Multiple-View Feature Modelling with Model Adjustment
Multiple-View Feature Modelling with Model Adjustment/Alex Noort
Thesis Technische Universiteit Delft. - With ref. - With summary in Dutch.
ISBN 90-9016343-3
NUR 980
Subject heading: Multiple feature views / Feature modelling

This work was carried out in graduate school ASCI.
ASCI dissertation series number 83.
Multiple-View Feature Modelling with Model Adjustment

PROEFSCHRIFT
ter verkrijging van de graad van doctor aan de Technische Universiteit Delft,
op gezag van de Rector Magnificus prof.dr.ir. J.T. Fokkema,
voorzitter van het College voor Promoties,
in het openbaar te verdedigen op dinsdag 26 november 2002 om 13:30 uur
door

Alexander NOORT
informatica ingenieur
geboren te Rijnsburg
Dit proefschrift is goedgekeurd door de promotor:
Prof.dr.ir. F.W. Jansen

Toegevoegd promotor: Dr. W.F. Bronsvoort

Samenstelling promotiecommissie:
Rector Magnificus, voorzitter
Prof.dr.ir. F.W. Jansen, Technische Universiteit Delft, promotor
Dr. W.F. Bronsvoort, Technische Universiteit Delft, toegevoegd promotor
Prof.dr. I. Horváth, Technische Universiteit Delft
Prof.dr.ir. M.J.L. van Tooren, Technische Universiteit Delft
Prof.dr.ir. B. de Vries, Technische Universiteit Eindhoven
Prof.dr. A. Bouras, IUT Lumiere-Université Lumiere Lyon 2
Preface

The research described in this thesis has been performed at the Computer Graphics and CAD/CAM group, of the Faculty of Information Technology and Systems of Delft University of Technology. Supported by my promotors, Erik Jansen and Wim Bronsvoort, I continued the research on feature modelling that has been going on in the group for several years now, and focused on multiple-view feature modelling to support conceptual and assembly design.

Parts of this research have been previously published in (Noort and Bronsvoort 1999; Noort and Bronsvoort 2000; Noort et al. 2000; Noort et al. 2001; Bronsvoort et al. 2001; Noort and Bronsvoort 2001a; Noort and Bronsvoort 2001b; Bronsvoort et al. 2002; Noort et al. 2002).

Many people of the Computer Graphics & CAD/CAM group supported me in a variety of ways. I wish to thank them all. In particular, I would like to mention the people of the feature modelling project.

Wim Bronsvoort, my supervisor, trained me to be a conscientious researcher; his suggestions and comments have been a valuable source of inspiration for me. I enjoyed to work with him, especially the many discussions we had, and hope to be able to continue the co-operation in some meaningful way.

Rafael Bidarra, my roommate and fellow researcher, provided a valuable source of feedback on various topics. His presence has broadened my view and his perseverance has been an example for me.

Ruud de Jong, the technical assistant of the project, also helped to assess the significance of new concepts with his critical remarks. Without his support, the current prototype modelling system would not have been possible.

Eelco van der Berg, who joined the project as a graduate student and continued as a PhD student on feature validation for freeform features, has been a nice and valuable colleague.

Further, I would like to thank the graduate students who have contributed to my PhD research project.

Geoffry Hoek has done a very good job in creating a GUI to integrate part and assembly design. Never satisfied with his own work, he kept refining it, resulting in a long but very valuable project.
Jeroen van der Berg has worked on a GUI to integrate conceptual design with part and assembly design. His dedication to his graduation project was unprecedented, and resulted in an important contribution to my research project.

I would also like to thank the members of my committee, in particular Prof. dr. I. Horváth, for the attention they paid to my thesis.

Finally, I would like to thank my parents, Jan and Joke, and my brother, Gerard, for their invaluable support.

Alex Noort
Delft, July 2002
## Contents

1 Introduction ........................................ 1
   1.1 Multiple feature views .......................... 2
   1.2 Automatic model adjustment ..................... 2
   1.3 Research context and objectives .................. 3
   1.4 Thesis overview ................................ 4

2 Approaches to multiple feature views ................. 5
   2.1 Design by features and feature recognition ........ 5
   2.2 One-way feature conversion ....................... 6
   2.3 Multiple-way feature conversion .................. 10
   2.4 Conclusions ................................... 13

3 Enhanced multiple-view feature modelling ............ 15
   3.1 Supported development phases ..................... 15
   3.2 Multiple feature views ......................... 19
   3.3 Part-oriented views ............................. 22

4 Conceptual design view ................................ 35
   4.1 Other approaches to configuration design ........ 35
   4.2 Model elements ................................ 41
   4.3 Model representation ............................ 45
   4.4 Model visualisation ............................. 45
   4.5 Model specification ............................. 48
   4.6 Conclusions ................................... 51

5 Assembly design view ................................ 53
   5.1 Other approaches to feature-based assembly design . 53
   5.2 Model elements ................................ 56
   5.3 Model representation ............................ 59
   5.4 Model visualisation ............................. 61
   5.5 Model specification ............................. 64
   5.6 Conclusions ................................... 67
<table>
<thead>
<tr>
<th>Chapter</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>Consistency maintenance</td>
<td>69</td>
</tr>
<tr>
<td>6.1</td>
<td>Use of different views</td>
<td>69</td>
</tr>
<tr>
<td>6.2</td>
<td>Conceptual design view and other views</td>
<td>73</td>
</tr>
<tr>
<td>6.3</td>
<td>Assembly design view and part detail design views</td>
<td>81</td>
</tr>
<tr>
<td>6.4</td>
<td>Conclusions</td>
<td>87</td>
</tr>
<tr>
<td>7</td>
<td>Automatic model adjustment</td>
<td>89</td>
</tr>
<tr>
<td>7.1</td>
<td>Model validation scheme</td>
<td>89</td>
</tr>
<tr>
<td>7.2</td>
<td>The principle of automatic model adjustment</td>
<td>92</td>
</tr>
<tr>
<td>7.3</td>
<td>Adjustment strategies for different types of constraints</td>
<td>95</td>
</tr>
<tr>
<td>7.4</td>
<td>Automatic model adjustment for feature conversion</td>
<td>102</td>
</tr>
<tr>
<td>7.5</td>
<td>Conclusions</td>
<td>105</td>
</tr>
<tr>
<td>8</td>
<td>Example modelling session</td>
<td>107</td>
</tr>
<tr>
<td>8.1</td>
<td>Initial conceptual design</td>
<td>107</td>
</tr>
<tr>
<td>8.2</td>
<td>Initial part design</td>
<td>111</td>
</tr>
<tr>
<td>8.3</td>
<td>Initial assembly design</td>
<td>112</td>
</tr>
<tr>
<td>8.4</td>
<td>Part design: refinement</td>
<td>113</td>
</tr>
<tr>
<td>8.5</td>
<td>Conceptual design: adding requirements</td>
<td>115</td>
</tr>
<tr>
<td>8.6</td>
<td>Assembly design: changing the model</td>
<td>116</td>
</tr>
<tr>
<td>9</td>
<td>Conclusions</td>
<td>119</td>
</tr>
<tr>
<td>9.1</td>
<td>Views</td>
<td>119</td>
</tr>
<tr>
<td>9.2</td>
<td>Consistency maintenance</td>
<td>120</td>
</tr>
<tr>
<td>9.3</td>
<td>Automatic model adjustment</td>
<td>122</td>
</tr>
<tr>
<td>9.4</td>
<td>Concluding statement</td>
<td>123</td>
</tr>
<tr>
<td>Bibliography</td>
<td></td>
<td>125</td>
</tr>
<tr>
<td>Index</td>
<td></td>
<td>133</td>
</tr>
<tr>
<td>Summary</td>
<td></td>
<td>137</td>
</tr>
<tr>
<td>Samenvatting</td>
<td></td>
<td>139</td>
</tr>
<tr>
<td>Curriculum Vitæ</td>
<td></td>
<td>143</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

Contemporary product development aims to design better products in less time. To realize this, concepts like concurrent engineering and feature modelling have been introduced. *Concurrent engineering* involves Design For X (DFX—where X stands for any product life cycle phase) and simultaneous engineering.

In *Design For X*, product quality is improved by taking into account requirements from several product life cycle phases during product development. Examples are Design for Assembly (Huang 1996; Boothroyd et al. 2002), taking into account assembly requirements, Design for Manufacturability (Huang 1996; Ulrich and Eppinger 2000), taking into account manufacturing requirements, and Design for Recyclability (Gatenby and Foo 1990; Huang 1996), taking into account recycling requirements.

In *simultaneous engineering*, product development time is reduced by enabling simultaneous activities in several product development phases (Bullinger and Warschat 1996). Traditionally, *sequential engineering* is used, which is often referred to as “over the wall engineering” (Hartley 1998), because the design that results from one product development phase is “tossed over the wall” to the next product development phase, without interaction between engineers working in subsequent phases. If the activities in the product development phases are done simultaneously, there can be useful interaction between the engineers and, as a result of this, the product development time can be considerably reduced.

*Feature modelling* is a way of product modelling that aims to provide ways to maintain more design intent in a product model than is possible in geometric modelling. The model of the product to be developed is built from features, which can be described as entities with attributes and properties that are useful in the reasoning process of one or more applications (Shah and Määttä 1995).

*Multiple-view feature modelling* is a concept that combines concurrent engineering and feature modelling. It supports applications from various phases of product development, by providing an own interpretation of, or view on, a product for each of these applications. Each *view* contains a feature model of the product specific for the corresponding application. Since a prerequisite for concurrent engineering is that the feature models of all views represent the same product, they have to be kept consistent (Bronsvoort and Jansen 1993).
1.1 Multiple feature views

In multiple-view feature modelling, each view has its own set of generic features, which are related to the associated product development phase, and are stored in the feature library of the view. The feature model of the view is built from instances of the generic features in this library.

Although a lot of research has been done on multiple-view feature modelling during the last years (see Chapter 2), there are still several major shortcomings of the current approaches. Two of these are that they focus on views for the later product development phases, in which the geometry of the product has to be fully specified, and that they deal with views for single parts only, whereas real products rarely consist of a single part.

In this thesis, a new approach to multiple-view feature modelling is described that overcomes these shortcomings. It supports high-level product design and design of products with multiple parts, thus also enlarging the capabilities to store design intent. It provides higher-level feature views for conceptual design and assembly design, in addition to the lower-level feature views for part detail design and part manufacturing planning.

Features are here defined as aspects of a product that have a functional meaning for some application. They relate different types of information on an aspect of the product, e.g. a connection feature in the assembly design view describes, among others, the freedom reduced between the components that are connected by the feature, and the regions of the geometry of these components that are involved in the connection.

Because the feature models of all views on a product should still represent the same product, they also have to be kept consistent. The process that makes sure that the feature models remain consistent under design changes is called consistency maintenance. Several new techniques have been developed to support this.

1.2 Automatic model adjustment

During the specification of the feature model of a view, it may occur that the feature model becomes invalid, or that a consistent feature model in some other view cannot be created. In such a situation, dimensions that are not critical for the function of the product may be adjusted in order to make the feature model valid or to make consistent feature models in all other views possible. Such dimensions may well exist in the product because it is, in general, not completely specified from the beginning (Mäntylä 1990; Ilies and Shapiro 1997; Gao and Roller 1998). In fact, some dimensions may always remain non-critical, especially those that are not related to a functional requirement.
A third major shortcoming of current approaches to multiple-view feature modelling is that they require the designer to manually adjust the feature model in such a situation. It can, however, be a tedious work for a designer to get a feature model that is valid and for which consistent feature models in all other views exist.

In this thesis, therefore also an approach is introduced that can automatically adjust a form feature model of a product in such a way that it becomes valid or that consistent feature models in the other views become possible. In this approach, the designer can mark dimensions in the feature model that are not critical, and this information is used to automatically adjust the model of the product.

### 1.3 Research context and objectives


The previous research projects resulted in a stand-alone feature-based approach for assembly modelling and planning, and a multiple-view form feature modelling system that maintains the consistency of the form feature models of all views and the validity of the form feature model of each separate view, and is based on volumetric form features. The multiple-view form feature modelling system provides a detail design view and a manufacturing planning view for a single part. So the results of these projects showed the shortcomings described in Sections 1.1 and 1.2.

The main goal of the research presented in this thesis is to overcome these shortcomings by developing a multiple-view feature modelling approach that supports the conceptual design phase, the assembly design phase, the part detail design phase and the part manufacturing planning phase of products, with facilities for automatic model adjustment. This involves the following research objectives:

- Develop views for the conceptual design phase and the assembly design phase that can be integrated with each other and with the existing views for the part detail design phase and part manufacturing planning phase.

- Develop a consistency maintenance approach to reflect changes that are made in the feature model of one view in the feature models of the other views.
• Develop an approach to automatically adjust an invalid form feature model or a form feature model for which a consistent feature model in another view cannot be created.

Implementation and evaluation of the new multiple-view feature modelling approach should be carried out within the prototype feature modelling system SPIFF (Bronsvoort et al. 1997) that resulted from the previous research.

1.4 Thesis overview

The remainder of this thesis is organised as follows.

Chapter 2 provides an overview of existing approaches to multiple feature views.

Chapter 3 introduces the new multiple-view feature modelling approach that has been developed, and describes the already existing part-oriented views.

Chapter 4 describes the conceptual design view, which supports the design of a product in terms of functional components and interfaces between them.

Chapter 5 describes the assembly design view, which supports the design of connections between components.

Chapter 6 describes the consistency maintenance of all feature models in the product model. It gives the consistency definitions for the different pairs of views, the algorithms that are used to check the consistency of the views, and the algorithms that are used to recover the consistency of views that are found to be inconsistent.

Chapter 7 contains a description of automatic model adjustment. It describes the concept and the techniques that are used to perform the adjustment.

Chapter 8 describes an example modelling session to illustrate the new techniques.

Finally, Chapter 9 presents the conclusions from this research.
Chapter 2

Approaches to multiple feature views

This chapter describes several approaches to managing multiple feature views. Section 2.1 describes design by features and feature recognition, two techniques that can, in principle, be used to build the feature models for multiple feature views, but have serious limitations in this respect. Section 2.2 describes one-way feature conversion, an approach to automatically keep the feature models of different views consistent, but with the restriction that the feature model of only one view can be edited. Section 2.3 describes multiple-way feature conversion, an approach to keep the feature models of different views consistent that allows the feature model of any view to be edited. Finally, Section 2.4 discusses the pros and cons of the described approaches to multiple feature views.

2.1 Design by features and feature recognition

In design by features, a designer constructs a model of a product using features. This requires a library of feature classes that can be instantiated by the designer. He can put his intent for regions of the product into the model of the product, and can use high-level operands and operations, such as slots and blends, which is considerably easier and faster than working with low-level geometric elements as in a solid modelling system (Shah and Mäntylä 1995).

In an environment that supports multiple design aspects or views, design by features can be used to separately create a feature model for each view. In this situation, the designer himself has to make sure that the feature models of the different views remain consistent, because these are independent from each other. If a change is made in the feature model of one view, then this change needs to be manually propagated to the feature models of the other views.

Feature recognition is defined as deriving a feature model from a given geometric model. Its main application has been in creating manufacturing process plans. Feature recognition can be performed either interactively by a designer, or automatically by a computer.
In *interactive feature recognition*, a designer creates features by interactively selecting topologic elements of a geometric model that make up a feature. An advantage of interactive feature recognition is that it is quite flexible, because the designer can designate any combination of topologic elements as a feature. However, a very serious disadvantage is that it is a laborious task to select all features in the model of a product by hand, in which errors can easily be made.

In *automatic feature recognition*, a computer application automatically generates a feature model, given a geometric model and some description of all possible features. Many different automatic feature recognition methods are available (Han et al. 2000). Some well-known methods are mentioned here. Graph-based methods recognise a feature in a geometric model by using a graph-matching technique to match, for example, an Attributed face Adjacency Graph (AAG) of a feature on the AAG of a boundary model (Ansaldi et al. 1985; Joshi and Chang 1988). Volume decomposition methods typically decompose the geometric model into a set of intermediate volumes, which are subsequently manipulated to produce features. Sakurai (1995) describes a cell-based decomposition method that first subtracts the model from a stock and subsequently decomposes the resulting volume into so-called atomic cells, combines these atomic cells into macro volumes, and finally classifies the macro volumes into features. Kim (1994) describes a convex-hull-based decomposition method that first subtracts the model from its convex hull and subsequently repeats this for the resulting volume and its convex hull, until the resulting volume is empty, and finally uses the found convex hulls to construct features. Geometric reasoning or hint-based methods recognise features by searching for a hint, i.e. something that is specific for a class of features in the boundary of a part, and subsequently constructing the largest feature related to that hint (Regli and Nau 1993; Vandenbrande and Requicha 1994). Because all these basic methods have advantages and limitations, several “hybrid” methods that combine aspects of the basic methods have been developed (Laakko 1993; Gao and Shah 1998; Li et al. 2000).

In an environment with multiple feature views, the feature model of each view can be recognised from the given geometric model. If a change is required, it has to be made in the geometric model, after which feature recognition has to be re-executed for each view. So directly changing a feature model is impossible here.

### 2.2 One-way feature conversion

*Feature conversion*, or, to be more precise, feature model conversion, is the process of deriving a new feature model for some application from an existing feature model for another application. An advantage of feature conversion over feature recognition is the possibility to use non-geometric information stored in the existing feature model to derive the new feature model.
In one-way feature conversion, typically, the feature models for the analysis and planning views are derived from the feature model of the design view. If the analysis or the planning results in the need to change the model of the product, then the changes have to be made in the feature model of the design view, and the feature models for the other views have to be updated from that model.

Cunningham and Dixon (1988) propose a multiple-view system architecture with one-way feature conversion. In this proposal, the designer uses design by features to build the feature model of the design view, called the primary view, and feature conversion is used to derive the feature models of, for example, the finite-element view, the process planning view and the assembly planning view, called secondary views (see Figure 2.1).

Anderson and Chang (1990) propose another approach to one-way feature conversion that mimics process planning procedures using geometric reasoning, to realize what they call feature refinement. Feature refinement converts design features into manufacturing features, and consists of three steps. First, the basic manufacturing features are generated by using a classification of the design features, which is based on the notion that the basic manufacturing features are a refinement of the design features. After that, the relations between the basic manufacturing features are determined. Such relations, e.g. patterns, nesting and intersection, are of importance for, for example, determining the order of manufacturing operations. Finally, basic manufacturing features are combined into more complex features that can be manufactured in one operation.
De Martino et al. (1994; 1998) describe a system for multiple-view feature modelling including one-way feature conversion. The feature model for each view is built from a view-specific library of features, which can be extended with user-defined features that can be built from existing features or from shapes. The feature models of the views are integrated by an intermediate model that contains, among others, a boundary representation of the product and a representation of the product by means of so-called shape features, i.e. generic protrusions and depressions. The shape features are extracted from the boundary representation of the product by the shape-feature recogniser, which uses a geometric-reasoning approach (see Figure 2.2). A new feature model for a view is created from the model with shape features by a so-called application-feature converter, which uses information from the feature library of the view to find features.

![Figure 2.2: Two depression shape features are identified, by first finding related faces in the boundary representation of the product, subsequently creating volumes from these faces, and finally storing these volumes as shape features in the intermediate model (after De Martino et al. (1994)).](image)

Suh and Wozny (1997) propose a multiple-view feature modelling system with one-way feature conversion in which features are defined as a subset of the boundary elements of a part model, i.e. faces, edges, and vertices, and their spatial relationships, i.e. constraints. New feature classes with a user-specified shape can be defined. A feature conversion algorithm is used to propagate the changes of a designer from the design view to the feature models of the application views. The feature conversion approach uses an intermediate model that is built from so-called Fundamental Features (FF), i.e. faces, edges and vertices, and Fundamental Spatial Relationships (FSR), such as Angle and Coplanar, that relate them. The feature model for an application view is recognised in the intermediate model by a hybrid feature recognition method that combines graph-based and hint-based techniques. The graph-based technique searches features in the intermediate model, given the constraint graph of a feature class, which is automatically built from the feature class definition (see Figure 2.3). In order to improve the performance of the graph-based technique, a characteristic FSR within a feature can be marked as a hint, and this FSR will be the first one to look for.
Bouras et al. (1999) describe a one-way feature conversion approach to create feature models of a product that are suitable for analysis applications. The approach first performs a morphological analysis on the shape of the product to find a set of shape features that have been applied to an initial gross shape. The morphological analysis first finds sets with planar, cylindrical, conic, toric and parametric faces, which describe the impact of the shape features on the gross shape. Subsequently it matches the defined impact of each possible shape feature class on these sets of faces to find the actual shape features in the product. Finally, it extracts the found shape features from the shape of the product and starts over again with the modified shape of the product. After all shape features have been found, features corresponding to a chosen view are generated from them. Then, the generated feature model is simplified, i.e. details that make the original design geometry too detailed and complicated for analysis purposes are removed, so that it will be easier to mesh and computationally less expensive to solve. Finally, it is idealised to obtain so-called working elements, which can be used in an analysis application.

The approaches discussed above only support feature conversion in a single direction, which is far from ideal. If, for example, a designer in an analysis view discovers that a dimension of the product has to be adjusted, then this adjustment cannot be made in that view, but has to be made in the design view. In the design view, however, the feature model is different from the one in the analysis view, and it may be quite difficult to find the right adjustments of the feature parameters in that view.

To allow the model of a product to be changed in the feature model of the view in which the need for it arises, multiple-way feature conversion has been developed.
The architecture described by De Martino et al. (1994; 1998) conceptually supports multiple-way feature conversion. However, they only allow users to create and update the feature model of the design view in order to avoid conflicts between updates from different views.

The technique described by Suh and Wozny (1997) also conceptually supports multiple-way feature conversion. However, the implementation only allows users to create and update the feature model of the design view, because feature models of the application views are allowed to represent only particular interesting regions of the part, and not the complete part, and thus cannot be the source for a conversion.

2.3 Multiple-way feature conversion

Several approaches that support multiple-way feature conversion, both conceptually and technically, will be described here.

The Computer Graphics and CAD/CAM Group of Delft University of Technology has been doing quite some work in the field of multiple-view feature modelling in the last decade, including multiple-way feature conversion. Bronsvoort and Jansen (1993) proposed multiple-way feature conversion as the ideal approach to multiple feature views, because it enables changes to the product model from an arbitrary view. They described an extension to the GeoNode feature modelling system (van Emmerik 1990) that would support multiple feature views and allow changes to be made in an arbitrary view. Later on, de Kraker et al. (1995; 1997), Bronsvoort et al. (1997), Dohmen (1998) and de Kraker (1998) describe a new approach to multiple-way feature conversion and its implementation in a prototype multiple-view feature modelling system called SPIFF. They define features as a physical part of an object that is mappable to a generic shape and has functional significance, store the feature classes for each view in a view-specific feature library, and allow new feature classes to be specified by the user (Bidarra et al. 1998a). In feature conversion, they distinguish opening a new view, in which the feature model of the new view is to be created, and changing an existing view, in which the feature models of the other views have to be updated. For opening a view, a geometric-reasoning-based feature recognition algorithm is used that derives the feature model of the new view from a non-manifold intermediate representation, called cellular model (Bidarra et al. 1998b) that will be described in Chapter 3. In order to propagate the changes made in the feature model of one view to the feature models of the other views, the feature models of the views are linked using a special kind of coplanar constraints, called link constraints (Dohmen et al. 1996); see also Figure 2.4, and Chapters 3 and 6. If these links fail because a constraint conflict occurs in a view, or the topology of the feature model of a view changes, incremental feature recognition is ap-
Multiple-way feature conversion

Figure 2.4: The faces of a feature in one view that are on the boundary of the product are linked to the faces of features in another view, in order to be able to propagate changes between the views.

plied for the changed region. A major advantage of this approach is that it also uses constraints to propagate changes between feature models, which is more efficient than using feature recognition only.

Jha and Gurumoorthy (2000a; 2000b) describe algorithms to automatically extract a feature model of a part from the geometric model of the part, and to propagate an adjustment in the feature model for one application to the feature models for other applications. Features are defined as a set of faces with distinct topological and geometric aspects, and the shape of features is limited to be a sweep solid, with end and shell faces. Feature classes are specified in terms of the characteristic arrangement of the faces of its instances that will overlap with existing faces of the model and the faces of its instances that will form new faces in the feature model (see Figure 2.5(a)). The algorithm to extract a feature model from a geometric model uses the characteristic topology of the faces of a particular feature class to find instances of that class. The algorithm to propagate adjustments is based on a data structure that incorporates the feature models for all applications. This so-called feature tree represents each feature model as a number of steps from one of the leaves of the tree to its root, each step introducing a new feature to that model (see Figure 2.5(b)). The algorithm first updates the root node of the feature tree for the changed feature, and subsequently descends each edge of the tree, updating the features found, until a node is found that is not influenced by the adjustment any more. A disadvantage of this approach is that it only supports features that are based on a (linear) sweep solid, and therefore features with, for example, a cone shape are not supported. Another disadvantage is that the approach is quite inefficient in propagating changes, since it does not exploit the relations between the different interpretations of the model in order to try to avoid the relative expensive feature extraction process.
Hoffmann and Joan-Arinyo (1998; 2000) propose an architecture in which the models for different applications are spread over different CAx systems, and a master model is used to associate the models. This architecture can perform multiple-way feature conversion between views. Each of the systems is said to be a client of the master model server, i.e. the process that manages the master model. A client can deposit some of its product information in the master model, but also associate private information with the information in the master model (see Figure 2.6). For example, a CAD client may deposit a boundary representation of the product to the master model, and a manufacturing planning application may associate private information with the elements of this boundary representation. The master model server informs the clients after a change of information in the master model with which they have associated private information, but the way the client reacts to this information is determined by the client itself. A major advantage of this approach is that it supports a product model to be distributed over different CAx systems, and therefore can support the integration of existing commercial applications. A disadvantage is that it only allows minor changes to be propagated back to the CAD client. Further, no implementation of the system has yet been reported.
2.4 Conclusions

The approaches to multiple features views described in the previous sections, which all use form features, share a number of shortcomings.

First, all approaches focus on the later product development phases, in which the geometry of the product has to be fully specified. They use the geometry of the product as a basis, and, therefore, cannot be applied in the early product development phases, such as conceptual design, in which the geometry has not to be fully specified yet. However, the early product development phases are also very important, because the choices made in those phases have an enormous influence on the resulting product.

Because in the early product development phases the geometry of the product is not completely known, the use of form features is limited here. For these phases, more abstract features need to be introduced.

The multiple-view feature modelling approach that is described in this thesis also provides a view to support the conceptual design phase. Chapter 3 gives an overview of the functionality of the multiple-view feature modelling approach to support conceptual design of the product. Chapter 4 describes the conceptual design view in detail, and Chapter 6 describes a way to keep the feature model of that view consistent with the feature models of the other views.

Second, all approaches deal with single parts only. Real products, however, rarely consist of a single part. Dealing with products that consist of multiple parts does not only involve dealing with the separate parts, but also involves dealing with the relations between the parts. In particular, maintaining the validity of all parts and the relations between them can be very complex.
The multiple-view feature modelling approach that is described in this thesis supports the design of products that consist of multiple parts, by allowing the designer to work on multiple parts at the same time using a detail design view and a manufacturing planning view for each part, by having the conceptual design view support multiple parts and relations between them, and by providing an assembly design view that takes into account multiple parts and the connections between them. Chapter 3 gives an overview of the functionality of the multiple-view feature modelling approach to support multiple parts and the relations between them. Chapter 4 describes the conceptual design view, Chapter 5 describes the assembly design view in detail, and Chapter 6 describes the integration of part-oriented and assembly-oriented modelling.

Third, all approaches discard the possibility that a feature model can become invalid, or that a consistent feature model in some other view cannot be created. However, these situations are inherent in keeping multiple feature views consistent and, therefore, must be supported by a feature conversion approach.

In such situations, the model of the product should be adjusted. To be able to adjust the model of the product automatically, the model should capture the intent of the designer in more detail than what is done in current multiple-view feature modelling approaches.

The multiple-view feature modelling approach that is described in this thesis supports the automatic adjustment of an invalid feature model, or a feature model for which a consistent feature model in some other view cannot be created. Chapter 7 describes the automatic model adjustment approach that is used for this.
Chapter 3
Enhanced multiple-view feature modelling

The enhanced multiple-view feature modelling approach described here focuses on the integration of aggregate-oriented modelling phases, which concentrate on the components in a product and the relations between them, with part-oriented modelling phases, which concentrate on a single part. This integration allows the feature models of the aggregate-oriented modelling phases to be kept consistent, but also the requirements from the part models to be kept satisfied in case an aggregate model is changed, and vice-versa. In the remainder of this thesis, enhanced multiple-view feature modelling will be referred to as multiple-view feature modelling.

Section 3.1 describes the four development phases that are supported in the multiple-view feature modelling approach. Section 3.2 describes the concept of feature views, which has been used before to support part-oriented modelling phases, and will be used here to support aggregate-oriented modelling phases. Finally, Section 3.3 describes the already existing feature views that support part detail design and part manufacturing planning.

3.1 Supported development phases

Four development phases are supported in the multiple-view feature modelling approach. The conceptual design phase and the assembly design phase support aggregate-oriented modelling; the part detail design phase and the part manufacturing planning phase part-oriented modelling.

The first phase that is supported is conceptual design. Two major movements in conceptual design exists; functional design and shape design. Functional design focuses on the function of a product, and includes dynamic, mechanical and electrical simulation approaches to create a product with all required functions. Shape design focuses on the shape of a product, and includes sketching approaches (Lipson and Shpitalni 2000) and configuration design approaches (Wielinga and Schreiber 1997).
Here, the focus is on conceptual shape design, in particular configuration design, since it best fits to the other supported development phases. It allows the designer to determine the configuration of a product by specifying components, which are to be implemented by one or more parts, and interfaces between them, which are to be implemented by a connection. The complete geometry of the components does not have to be specified.

Components are specified by means of a base shape with shape concepts and references on it. The base shape is used to attach and position shape concepts and references, and gives an impression of the shape of the component. Shape concepts are used to specify functional requirements on the geometry of a component, e.g. that there should be a depression, possibly with a cylindrical shape, somewhere on the component. References are used to position interfaces, i.e. connections, between components, shape concepts and other references. Interfaces between components are specified by means of Degrees Of Freedom (DOFs) between references on the components.

An example of a model of a modern version of the historical high-wheel bicycle (Figure 3.1(a)) that results from the conceptual design phase is given here. The model represents the components of the product, with their important aspects only, and the interfaces between the components. Figure 3.1(b+c) show some images of the model. Conceptual design will be elaborated in Chapter 4.

![Figure 3.1](image-url)

**Figure 3.1**: The historical high-wheel bicycle (a), and some images of the model that result from the conceptual design phase of a modern version of it: the base and the shape concepts of the front fork component (b), and the interfaces between the components (c).
The second phase that is supported is *assembly design*. Many approaches to assembly design exist. Some support all phases of product design that involve multiple parts, and provide several tools to analyse, for example, whether the components can be assembled (Zha et al. 2001). Others focus on a specific aspect, such as the mate and contact relations between components (Whitney and Mantripragada 1998), ways to represent complex assemblies (Heissermann and Mattikalli 1998), or the use of features to represent information on the way components are connected (van Holland and Bronsvoort 2000).

The approach to assembly design that is supported here focuses on connection design. It allows the designer to specify the type of connection between components and the geometry for the connection on the components. In this phase, in fact the interfaces that have been specified in conceptual design are refined.

A component in assembly design can be a single component, which represents a single part, or a compound component, which represents two or more connected sub-components. The geometry of components only needs to be specified as far as it is involved in connections. Connections are specified on regions of the connected components, such as a rib or a slot, and represent a reduction of the DOFs between the components.

An example of an assembly model of the high-wheel bicycle that results from the assembly design phase is given in Figure 3.2. The example shows the connection of the rear wheel to the rear fork, with an additional axis component that is connected to the rear fork with a fixed pen-hole connection and to the rear wheel with a rotating pen-hole connection. Assembly design will be elaborated in Chapter 5.

Figure 3.2: The model of the assembly that consists of the rear wheel, the rear fork and the rear axis component of the bicycle, as it results from the assembly design phase.
The third phase that is supported is part detail design. Different approaches to part detail design focus on different aspects of a part, such as shape, tolerances and strength.

The approach to part detail design that is supported here focuses on the shape of parts, and allows the designer to refine the shape of a component as it has been specified in conceptual design. The shape of the part is specified by means of high-level model elements, such as corner protrusion and passage, which are related by means of constraints, such as distance and parallel constraints.

The model in the detail design view on a part of the high-wheel bicycle is given in Figure 3.3. The example shows the detail geometry of the part that is used to fix the saddle to the frame.

![Figure 3.3: The model of the part that is used to fix the saddle as it results from the part detail design phase.](image)

The fourth phase that is supported is part manufacturing planning.

The approach to manufacturing planning that is supported here focuses on the type of the manufacturing operations that are needed to create a part, taking into account the capabilities of the available manufacturing equipment.

The model in the manufacturing planning view on the part of the high-wheel bicycle from Figure 3.3 is given in Figure 3.4. The model represents the regions of a stock of material that have to be removed by manufacturing operations, in order to obtain the part geometry as it has been specified in the part detail design phase.

Part detail design and manufacturing planning are elaborated in Section 3.3.

![Figure 3.4: The model that results from the manufacturing planning phase for the model of Figure 3.3.](image)
3.2 Multiple feature views

To support the product development phases described above, the concept of a view on a product is used. It consists of a feature model of the product, and libraries of feature classes of which instances are created to build the feature model. Each development phase has its own view on a product or part of a product, which represents only those aspects that are relevant for that phase (Bronsvoort et al. 1997).

Feature classes

The feature classes in the libraries of a view store pre-defined, parametrised, functional aspects from the product development phase that is related to the view. Classes have to be specified only once, and instances of them can subsequently be used in the feature model of this type of view on any product.

Feature class specification involves specification of feature elements, validity conditions, and the interface to the designer.

The feature elements of a feature class are the constituents of the feature. Feature elements can be simple elements such as a face or an algebraic variable, but can also be complex elements such as a form feature.

The validity conditions of a feature class relate the feature elements of the class in such a way that they form a valid functional unit. They are specified by constraints on the elements, and a feature is said to be valid if all constraints are satisfied. Constraints can be geometric constraints, which specify relations between feature faces, such as parallel and distance-face-face, algebraic constraints, which specify relations between algebraic variables, such as equal and plus, dimension constraints, which specify the value of an algebraic variable to be within a given range, interaction constraints, which include constraints that disallow a certain type of geometric overlap between feature shapes, and boundary constraints, which specify the extent to which feature faces should be on the model boundary.

The interface of a feature class determines how the designer can interact with the instances of the class. It specifies the data that has to be specified in order to create an instance of the feature. For example, a certain type of connection may require two components, one form feature on each component, and a dimension to create the connection. A protrusion may need a face to attach it, two other faces and two values to position it on the attach face, and values for its height, width and length.

Feature model

The feature model of a view can be specified from instances of the feature classes in the feature libraries of the view, and, depending on the view, some additional entities, such as model validity conditions.
A feature model can be visualised using several types of cameras. The cameras use geometry, icons and text to show the shape of the product that is represented by the feature model, the functional information in the feature model and the structure of the feature model. Most types of cameras have parameters that specify the way the camera shows the feature model, which can be adjusted by the user to optimise the picture shown by the camera for a given situation. Examples of cameras are given in Figure 3.5.

![Figure 3.5: A geometry camera showing the geometry of a product and some reference surfaces (a), and a relational graph camera showing the relations between the components of the same product, denoted by icons (b).](image)

A feature model should always remain valid, i.e. all validity conditions in the feature model should always remain satisfied. Feature validity maintenance is the process that ensures this. It checks the validity of the model after new model elements have been added, after the values of parameters of existing model elements have been changed, and after a model element has been removed, and it ensures that the validity is recovered if the model has become invalid (Bidarra and Bronsvoort 2000).

The validity of the feature model can be recovered by either adjusting values of parameters of a model element in such a way that the model becomes valid, or by removing a model element such that the model becomes valid.

**Views**

In the multiple-view feature modelling approach, a product model consists of the feature model of the view that supports the conceptual design phase of the product, the feature model of the view that supports the assembly design
phase of the product, and feature models of the views that support the detail design phase and the manufacturing planning phase of each part in the product (see Figure 3.6).

The views can be divided into a group of views that deal with all components of a product and the relations between them, i.e. the aggregate-oriented views, and a group of views that deal only with an elementary component of an aggregate, but no relations, i.e. the part-oriented views. The conceptual design view and the assembly design view are the aggregate-oriented views, and the part detail design views and the part manufacturing planning views are the part-oriented views.

**Consistency maintenance**

The product model in the multiple-view feature modelling approach combines the feature models of all views on the product, each of which represents the aspects of the product that are relevant for its associated product development phase. However, feature models of different views may represent the same aspect, and inconsistencies between the feature models can occur if such aspect is specified differently in two views.

Whether or not the feature models of two views are consistent is determined by the consistency conditions that have been defined between their elements. The feature models of views are consistent if all consistency conditions are satisfied. Additional consistency conditions have to be defined if a new feature view is introduced. Examples of consistency conditions are that the pairs of connected components from two views that are associated should have the same degrees of freedom and that the feature models of two views should represent the same geometry.
Consistency maintenance is used to make sure that the feature models of the views remain consistent after a model operation in one of the views. It consists of consistency checking, which is used to check whether the feature models of two views are still consistent, and consistency recovery, which is used to make the feature models of two views consistent again if they have been found to be inconsistent.

The multiple-view feature modelling approach uses multiple-way feature conversion to recover the consistency of feature models of views. In case a new view is opened, the feature model of that view is obviously inconsistent with the feature models of the existing views, and the consistency is recovered by either automatically or manually deriving the feature model for the view from the feature models of the existing views. In case the feature model of an existing view is changed and the feature models of some views have been found to be inconsistent, the feature models of these views are updated in order to recover the consistency.

3.3 Part-oriented views

The part detail design view and the part manufacturing planning view are two part-oriented views that have been developed by Bronsvoort et al. (1997) to support the part detail design phase and the part manufacturing planning phase, respectively. They will be used in the multiple-view feature modelling system to support part-oriented modelling. The feature models of both views are built from form features, model constraints, and references. Form features have been defined as regions of the geometry of a product that have some functional meaning.

Major parts of the work described in this section draw on work by de Kraker (1998), Dohmen (1998) and Bidarra (1999). Some minor parts, in particular on the use of references and of the feature dependency graph in consistency maintenance, are new.

Feature classes

The feature libraries of these views can be managed by using a feature library manager, which allows new form feature classes to be specified, existing classes to be changed and obsolete classes to be removed (Bidarra et al. 1998a). New classes are specified in three phases.

In the first phase, the shape of the feature class is specified, including shape faces and geometric validity conditions within the shape. The shape can be either a predefined base shape that includes all necessary shape faces, a dimensioning scheme and geometric validity conditions between the faces, or
a user-defined compound shape that includes several base shapes, a dimen-
sioning scheme, additional user-defined variables, additional geometric and 
algebraic validity conditions, and a naming scheme for the external faces. Ex-
amples of a base shape are trapezoidal block and cone (see Figure 3.7); an 
example of a compound shape is the I-shape (see also Figure 3.7). Examples 
of geometric validity conditions are parallel, coincident, distance and angle; 
examples of algebraic validity conditions are plus and equal.

Figure 3.7: The trapezoidal block and the cone shape are base shapes, the I-shape is 
a compound shape that consists of a block and four cylinder shapes.

In the second phase, validity conditions that are not required for the shape 
can be specified. There can be attach constraints that specify the way the shape 
of a feature is attached to the model, dimension constraints that specify a di-
mension of the feature to be within a given range, boundary constraints that 
specify a feature face to be partly or completely on the boundary of the prod-
uct or not, and interaction constraints that specify, for example, that no addi-
tive feature may overlap with a given subtractive feature. All except the attach 
constraints can also be used as model constraints, i.e. model elements that spec-
ify additional validity conditions for a single or a set of feature instances.

In the third phase, the positioning and orientation scheme of the feature 
class is specified. In general, an instance of a feature class is placed in a model 
by attaching it to existing feature faces in the model and subsequently specify-
ing its position and orientation by giving the distance and angle with respect 
to existing feature faces.

As a consequence of this declarative way of specifying feature classes, each 
feature class has an explicit feature class interface, i.e. a description of the way 
the designer can interact with an instance of the feature class. This describes 
the parameters, i.e. the data that has to be specified to create an instance of the 
feature class (Bidarra and Bronsvoort 2000b).

The feature library manager stores the feature class definition as a class in 
LOOKS, an object-oriented imperative programming language (Peeters 1993) 
that can be interpreted by the multiple-view feature modelling system.

The feature classes in the feature library of the part detail design view pro-
vide the designer with the functionality to specify the detail geometry of a 
part. These feature classes represent requirements on the product that are not
associated with the application of any other view. Examples of these feature
classes are through hole, i.e. a cylindrical hole with both the top and the bot-
tom face as entrance face, and rectangular corner protrusion, i.e. a protrusion
with block shape of which a corner is aligned with the corner of another fea-
ture.

The feature classes in the feature library of the manufacturing planning
view represent the operations of some “classical” manufacturing machines,
such as drilling and milling machines, and are therefore subtractive. Exam-
ples of these feature classes are blind hole, i.e. a depression with cylindrical
shape and only one entrance face, which can be manufactured using a drilling
operation, and step, i.e. a depression with block shape and four entrance faces,
which can be manufactured using a milling operation.

References

References allow the designer to specify points, lines, surfaces and arrows that
are not part of the geometry of the model, but can be used to position and
orientate other model elements. They will be described here as separate model
elements, called model references, but can also be included in feature classes.

The references classes that will be described here were inspired by
the references in other research systems, such as the “nodes” in Geo-
Node (van Emmerik 1990), and in commercial modelling systems, such as
“datum planes” in Pro/Engineer (Parametric Technology Corporation 2002).

However, the reference classes described here provide a way to encapsu-
late the constraints that are used to position a reference, which eases the use
of references, and are stored in an extendible library. New reference classes
can be added, existing classes can be changed, and obsolete classes can be
removed.

A reference consists of a geometry, a positioning and orientation scheme,
and an interface. The geometry of the reference can be a single point, line,
surface, or arrow, or a combination of such elements with their relative posi-
tion and orientation specified by constraints. The positioning and orientation
scheme places the reference within the model and consists of geometric con-
straints. As a result, reference classes also have an explicit interface, which
specifies the data that has to be specified to create an instance of the class.

An example of the benefit of the use of references to position two pocket
features will be given here, based on a model of a crown part of a bicycle, i.e.
the part that connects the legs of the front fork. Because the pocket features
are used to fix the legs, they should be symmetrically positioned to ensure sta-
bility of the bicycle and they should be 15 cm apart to allow space for the front
wheel between the legs. Without references, the pockets would need be posi-
tioned and dimensioned with a scheme similar to that of Figure 3.8(a) in order
to make sure that the requirements are met. In this less-intuitive scheme, the
size of the base, the size of the pockets and the values for the positioning con-
straints, all influence the fact whether the pockets are placed symmetrically.
With parallel-surface references, i.e. the bold lines in Figure 3.8(b), the pockets
could be positioned and dimensioned with the scheme of Figure 3.8(b), which
is much more intuitive.

![Figure 3.8: Positioning two pockets without references (a) is usually less intuitive than
positioning them using references (b).](image)

However, the positioning scheme of Figure 3.8(b) could be further simplified if
the pocket features would contain a surface of symmetry, because this surface
could then be used to position the pockets. To represent the surface of sym-
metry, a reference could be added to the feature class; it could be positioned
with respect to the faces of the feature.

An example of the use of the surface of symmetry is given in Figure 3.9(a). Each
pocket form feature is positioned with respect to two model references,
using one of its side faces and its own reference surface (see Figure 3.9(b)).

![Figure 3.9: Positioning pocket features with model references and an own reference is
even more intuitive.](image)
Model representation

The feature model of a part view is represented by a feature dependency graph, and a cellular model that is shared with the feature models of the other part view and the assembly design view.

The feature dependency graph represents all feature instances of the feature model of a view, each of them with its set of elements (e.g. shape, parameters and constraints), model references and model constraints. These model elements are interrelated by dependency relations, in such a way that no disconnected subgraphs exist. A dependency relation appears if an entity is attached, positioned, or in some other way related to another entity. Further, an entity is said to depend on another entity if a dependency relation exists between them, or a dependency relation exists with yet another entity that depends on the other entity (Bidarra and Bronsvoort 2000b). An example of the feature dependency graph of the product of Figure 3.9 is given in Figure 3.10.

![Feature Dependency Graph Example](image)

Figure 3.10: An example of a dependency graph with features, model references and a model constraint.

In this way, the feature dependency graph provides a high-level interface to the feature model. Interaction between the user of the modelling system and the model takes place in terms of entities in the feature dependency graph, and all modelling computations are primarily carried out at this level. Each entity in the feature dependency graph may be queried about its current parameter values, and the features and model references may be queried about their global position and orientation.

The feature dependency graph contains no evaluated geometry, but does contain all information necessary to generate and maintain this in the cellular model (Bidarra and Bronsvoort 2000b).
The cellular model is a non-manifold representation of the feature model geometry, integrating the contributions from all features in the feature dependency graph. It represents the geometry of a part as a set of volumetric cells of arbitrary shape that do not geometrically overlap, in such a way that each one either lies completely inside the shape of a feature, or completely outside it. The decomposition of the cellular model into cells is determined by the overlaps between the shapes of the features in the feature model, i.e. interactions between feature shapes introduce additional cells (Bidarra et al. 1998b; Bidarra and Bronsvoort 2000b).

Each cell contains information on the features whose volume overlaps with the volume of the cell, and each cell face contains information on the feature faces that overlap with it. In addition, a cell also contains information on the fact whether its volume represents material, i.e. the cell has additive nature, or not, i.e. the cell has subtractive nature; a cell face also contains information on the fact whether it represents part of the boundary of the feature model, i.e. has material on one side and no material on the other side, or not.

This information is stored by means of owner lists. Each cell and cell face has such an owner list, which indicates to which features and feature faces it belongs in each view (Bidarra et al. 1998b). The nature of a cell is determined by the features in that view that overlap with the cell and the dependencies between them (Bidarra and Bronsvoort 2000b). Figure 3.11 shows the cellular model for the part of Figure 3.9, in particular the cells that originated from the overlapping chamfers.

![Figure 3.11: A cellular model with owner lists of some cells that involve overlapping chamfers.](image)

**Model visualisation**

The feature models of the part detail design view and the part manufacturing planning view are visualised using geometry cameras and graph cameras.
A *geometry camera* uses non-photorealistic rendering to give better insight into the specific feature geometry in the model. Non-photorealistic rendering, here, aims to generate images of objects that enhance the perception of the objects instead of photo-like images, just like human illustrators do in technical illustrations. By selectively shading certain regions of the geometry, and by adding, for example, silhouettes and hidden lines to shaded images, the perception of the objects by the user can be considerably improved (Forrest 1986; Saito and Takahashi 1990; Gooch et al. 1998).

Geometry cameras can visualise the feature models of the part detail design view and the part manufacturing planning view in many different ways. They can visualise the model as a line image, with the hidden lines removed or dashed as option, as shaded image, or as hybrid image, with lines superimposed on the shaded image. In addition, they can visualise a selected subset of features in a different way, e.g. the model with a line image and the selected features shaded in the colours that have been specified for their feature class, and they can even visualise the faces of the selected features that are not on the boundary of the model (Bronsvoort et al. 2002).

Examples of the different ways geometry cameras can show the geometry of a feature model are given here. Figure 3.12(a) shows a camera on a feature model of the part detail design view, in which the features that will be used to connect the part to another part are shaded, and the rest of the model is visualised with dashed hidden lines. Figure 3.12(b) shows a camera on the corresponding feature model in the part manufacturing planning view, in which the tool-entrance faces have been shaded, and the rest of the model is again visualised with dashed hidden lines.

![Figure 3.12](image_url)

**Figure 3.12:** A geometry camera can show different aspects of the geometry of a feature model.

A *graph camera* shows the elements in the feature model of a view, and the way they are related, by means of a graph. In the graph, the form features and model references are visualised as a node with a label or an icon that shows a symbolic representation of the class of the model element. The constraints
that are used to attach form features and to position form features and model references with respect to other elements are visualised as directed arcs from the node that represents the form feature or model reference to the nodes that represent the other elements. Model constraints are visualised as a node with a label or an icon that shows a symbolic representation of the class of the model constraint, and directed arcs from this node to the nodes that represent the model elements on which the constraint has been specified.

Figure 3.13 shows the graph camera for the model of Figure 3.9. It shows the node that represents the base feature of the model in the middle of the bottom row, the nodes that represent the reference surfaces that are used to position the blind hole and the pocket features in the middle row, the nodes that represent the pocket features at the top row, the node that represents the blind hole feature in the middle row, the nodes that represent the chamfer features at the bottom row, and the node that represents the model constraint between the two pocket features in the top row.

**Model specification**

Feature instances, model references and model constraints can be added to the feature model of either the part detail design view or the part manufacturing planning view, the values of the parameters of existing model elements in the feature models of these views may be changed, and model elements from the feature models of these views may be removed.
New model elements are added to the model by specifying the class, and values for the required parameters. In contrast to most other modelling systems, only constants and elements of existing features may be used as value for the parameters of a feature, elements of the boundary representation of the product may not (Bidarra and Bronsvoort 2002). The advantage of this approach is that all model elements are independent of the elements of the boundary representation, and therefore do not unexpectedly change as result of a change of the boundary representation, as frequently happens in other modelling systems (Raghothama and Shapiro 1998).

An example of adding a model element to a model is given, in which the blind hole of the model of Figure 3.8 is created. Figure 3.14(a) shows the modelling panel, which is used to create the blind hole feature that is attached to the bottom face of the base, and positioned with respect to two parallel reference surfaces in the model of Figure 3.14(b). Figure 3.14(c) shows the resulting model.

![Figure 3.14: Adding a blind hole feature (a) to a model (b) results in the adjusted model (c).](image)

The values of the parameters of existing model elements in either part detail design or part manufacturing planning view may be changed in order to change the feature models of both views. In contrast to most other
modelling systems, in which model elements may only be changed in such a way that they keep referring to model elements that have been created before them, here it is allowed to change the value of the parameters of model elements to refer to any other model elements. To support this, the system uses the actual dependencies between the model elements, instead of the order in which the model elements have been created, to evaluate the geometry of the model (Bidarra and Bronsvoort 2000b).

Features can only be removed from the model if no other model element depends on it. Model elements that depend on a feature refer to faces or dimensions of the feature. Removing such feature would invalidate these references and should therefore be avoided.

Of course, neither of these operations should make the feature model of either view invalid, or the feature models of the views inconsistent. The next two subsections describe the maintenance of the validity in each view, and the maintenance of the consistency of the views.

**Validity maintenance**

During the specification of the feature model in one of the views, *validity maintenance* is performed to make sure that the validity conditions of all existing features in the model remain satisfied. After each operation, the validity of the feature model is checked, and if the model has become invalid, the validity has to be recovered.

*Validity checking* is done in three phases (Bidarra and Bronsvoort 1999; Bidarra and Bronsvoort 2000b). First, in case of a remove operation, it is checked whether the model element to be removed has dependent model elements. If so, the model has become invalid. Second, if the operation is not a remove operation, the geometric and algebraic constraints are solved in order to determine whether these constraints can be satisfied. If some geometric or algebraic constraint cannot be satisfied, the model has become invalid. Third, the dimension, boundary and interaction constraints are checked in order to determine whether these constraints are satisfied. If some dimension, boundary or interaction constraint is found that is unsatisfied, the model has become invalid.

If the model has become invalid, the validity has to be recovered.

The validity of the feature model can be recovered either by adjusting values of the parameters of a model element in such a way that the model becomes valid, or by removing a model element such that the model becomes valid (Bidarra and Bronsvoort 1999; Bidarra and Bronsvoort 2000b). Automatic recovery of invalid feature models will be presented in Chapter 7. If automatic recovery is not possible, manual recovery is needed. This will be shortly described here.
In manual recovery of invalid feature models, the designer has to specify modelling operations that make the model valid again. He can make use of automatically generated hints for this. The hints that are generated depend on the phase of the model validity checking process in which an invalid situation has been detected. If an invalid situation is detected during the checking for model elements that depend on a model element to be removed, then the dependent model elements are presented to the designer, and it is suggested to change the values of their parameters or to remove them from the model. If an invalid situation is detected during the solving of the geometric and algebraic constraints, the model elements that caused the invalid situation are presented to the designer, and it is suggested to change the values of their parameters. If an invalid situation is detected during the checking of the dimension constraints, the associated parameter, its model element and the admissible range for that parameter are presented to the designer, and it is suggested to change the value of the parameter in such a way that it will be within the admissible range. Finally, if an invalid situation is detected during the checking of the boundary and interaction constraints, the causes and the effect of the detected situation are presented to the designer, and depending on the type of situation, specific corrective actions are suggested (Bidarra and Bronsvoort 1999; Bidarra and Bronsvoort 2000b).

Consistency maintenance

Consistency maintenance is performed to make sure that the consistency conditions between the feature models of different views become and remain satisfied. The consistency condition for the feature models of the part detail design view and the part manufacturing planning view is that they should represent the same geometry (de Kraker et al. 1995).

In case the feature model of one of the two views is changed, a linking scheme is used to try to propagate the change to the feature model of the other view, in order to avoid the feature model of the other view to become inconsistent. The linking scheme is a special constraint management scheme, which uses links between the feature models of both views, to dimension and position the features of the target view based on the features of the source view.

Links are constraints that specify two feature faces of the two views to be co-planar. Links are automatically created by the system between feature faces that represent the same model face. The feature faces in both views that represent a model face can be found in the owner list of that model face in the cellular model. An example of a subset of the links between the feature models of the part detail design view and the part manufacturing planning view on a part is given in Figure 3.15. If the feature model of one of these views is changed, i.e. if one or more of the feature faces that are on the boundary of the model are moved, then the links propagate this change to the feature model of the other view.
Part-oriented views

Figure 3.15: Feature faces of the part detail design view (a) are linked with overlapping feature faces (b) of the part manufacturing planning view (c).

To allow the links to dimension and position the features in the target view, the constraints that specify the dimension and position of the features in that view are deactivated. The dimension and position of these features is then determined by the dimension and position of the features in the source view, via the links between faces of the features in the source and target view, and the attach relations between the features in the target view. In this way changes to the dimension and position of a feature in the source view can automatically be propagated to the features in the target view (Dohmen et al. 1996).

After the model has been changed, consistency checking is performed in order to check whether the consistency condition is still satisfied. It is satisfied if all cells in the cellular model have the same nature for both views. If a view has been found to be inconsistent with the view that is changed, its consistency has to be recovered.

When a new view is opened, the consistency condition is always violated, and the feature model of the new view has to be made consistent with the feature models of the existing view (de Kraker et al. 1997).

Consistency recovery removes existing features from, and adds new features to the feature model of the target view, in order to make it consistent with the feature model of the source view.

A feature is removed from the feature model of the target view if it has become invalid or if it intersects with a cell that has a different nature in the target view than in the source view. After all necessary features have been removed, feature recognition is used to search new features to be added to the model of the target view in order to make it consistent.
The order of feature classes to search instances of, is determined by the search strategy that is specified for the view. If the search strategy specifies a certain feature class to be looked for, feature recognition is used to find all possible instances of the class in the current model, and the largest instance is added to the model (de Kraker et al. 1997).
Chapter 4

Conceptual design view

The conceptual design view supports the conceptual design of a product. In particular, it supports configuration design, i.e. the specification of the way the product is built from components and interfaces. Components contain, among others, shape concepts, which specify a functional requirement for a region of the component geometry, and references, which can be used to specify interfaces between components. Interfaces specify the reduction of the degrees of freedom between components. The relations between the conceptual design view and the assembly design view, the part detail design and part manufacturing planning views are described in Chapter 6.

Section 4.1 describes other approaches to configuration design. Section 4.2 presents the elements of the feature model of the conceptual design view. Section 4.3 describes the way the feature model is represented. Section 4.4 presents the different types of cameras that can be used to visualise the feature model. Section 4.5 describes the way the feature model can be specified using its model elements. Finally, Section 4.6 gives some conclusions.

4.1 Other approaches to configuration design

Four existing modelling systems that support configuration design will be described.

Van Emmerik (1990) describes the GeoNode system, which supports top-down product development for conceptual design, including configuration design, and part detail design. In conceptual design, the system uses a hierarchical design scheme, in which the product is represented by assemblies, sub-assemblies and components. Each model element needs an initial geometric representation in order to be able to generate a graphical preview. Assemblies are specified by means of the topologic and geometric organisation of sub-assemblies and components.

The topologic organisation of an assembly specifies the way the sub-assemblies and components of the assembly are connected. An example of the topologic organisation of the sub-assemblies and parts of a cart is given in Figure 4.1, which shows a frame sub-assembly with a front and a rear axle, two wheel sub-assemblies connected to the front axle, and two wheel sub-assemblies connected to the rear axle.
The geometric organisation of an assembly specifies the dimensions of the assembly. Examples of the geometric organisation of the assembly in Figure 4.1 are the diameter of the wheels and the distance between the front and rear axle.

After the topologic and geometric organisation of an assembly have been specified, sub-assemblies can be refined into multiple sub-assemblies and components. This continues until all sub-assemblies have been split into components.

After the conceptual design phase, the components have to be materialised. For this, either standard components can be chosen from a library or, alternatively, new components can be created bottom up from geometric primitives, features and other generic objects. An example of the materialisation of the frame component of the cart is given in Figure 4.2.

The GeoNode system allows multiple alternative materialisations to be made for each component, in order to compare them and choose the optimal solution. An example of two possible materialisations for the cart of Figure 4.1 is given in Figure 4.3.
Horváth et al. (1995) describe a conceptual design methodology that supports a designer to specify the so-called organ structure of a mechanical product, and has been implemented in their PRODES system. An organ structure describes the functional components and the connections between the components. Each organ in an organ structure has a fixed skeleton to represent the geometry and morphology in an abstract way. A skeleton consists of ports, which represent a point that is related to another skeleton, contact surface patches, which represent the morphological characteristics of the contact point of a port, and bones, which connect the ports of a skeleton. The relations between the skeletons result in an abstract morphological model, by which the product can be represented satisfactorily in the early stages of design.

Concept feature objects are provided to specify both the functional and morphological aspects of the organ structure. They capture and model concepts that are known and used by designers during the conceptualisation process, and exist at three levels of abstraction: abstract, concrete and instance concept feature objects. An abstract concept feature object has a varying number of ports, bones and contact surfaces. A concrete feature object is derived from an abstract concept feature object by specifying the necessary ports, bones and contact surfaces, and has a fixed number of parameters. Finally, an instance concept feature object is derived from a concrete concept feature object by specifying the values for the parameters.

An example of a skeleton model of the crank mechanism of an internal combustion engine is given here. It consists of a crankshaft and a piston, which are connected by a rod (see Figure 4.4). The ports are shown by means of their contact surfaces.

Guan et al. (1997) describe the GEMCON system, which allows a designer to incrementally model geometric configurations, by allowing vague, along with precise, geometric information, and the gradual evolving of configurations into concrete and precise models. An important aspect of the system is that it adheres to the minimal commitment principle, i.e. it does not require the designer to specify aspects of the model that are not relevant yet.
The representation used by the system consists of a parameterised geometric model, which provides the parametric representation of the geometric properties of each object, a geometric structure model, which provides the required elements for representing a geometric configuration consisting of objects at different levels of detail, and several constraint models, which capture the constraints that define and relate the geometric properties of the objects.

The parameterised geometric model is built from geom objects, i.e. elements with a shape, size, location and orientation. The value of a size parameter can be defined either precisely or approximately, and is represented by an interval. The location of an object is characterised by a datum point, which lies in an orthogonal 3D cubic uncertain region that captures the approximation or uncertainty associated with the location. An example of the representation of an object with a cuboid shape is given in Figure 4.5.

Figure 4.4: An example of a skeleton model (after Horváth et al. (1995)).

Figure 4.5: Parameterised representation of an object with cuboid shape. (after Guan et al. (1997)).
The geometric structure model represents the different levels of detail for an object by a directed acyclic graph that represents the geoms that refine a geom as children of the node that represents that geom.

Different constraint models are used to hold the constraints that specify the size, orientation and location of a geom object. The size of an object can be specified by independent size constraints, such as inequality, range and equality. The orientation of the object can be specified by orientation constraints that specify the rotation of objects in multiples of 90 degrees around the coordinate axes. The location of the object can be specified either by point constraints, which specify the location with respect to the world coordinate system, or spatial relations, which constrain the location to be, for example, above or left of another object.

Csabai et al. (1998) describe a Layout Design Tool (LDT), which facilitates the capturing of the human intent during the conceptual design. This system also adheres to the minimal commitment principle.

In this system, the product is represented by an abstract representation called product layout, which represents the structure and arrangement of the components of the product. The basic entities of the product layout, called design spaces, are used to dispatch parts of the product to the designers working on the product. The connections between the design spaces are represented by interface features.

Design spaces are polyhedral objects, whose shape should be sophisticated enough to reflect the component inside, but simple enough to allow them to be created and handled rapidly. The shape can either be chosen from a library of frequently used shapes, such as rectangular boxed and L-shaped areas, or can be built by the designer from faces and constraints. An example of a design space is given in Figure 4.6.

![Design space for a simple component](image-url)

Figure 4.6: Design space for a simple component. Note that some part of the final component is outside the design space (after Csabai et al. (1998)).
Interface features are intended for use as references for constraints that specify relations between design spaces, and to position connection form features. Because an interface feature is a reference for a constraint between design spaces, the designer of a design space is not allowed to modify it.

High-level and low-level constraints can be used to relate the design spaces. High-level constraints are constraints that represent joints with related degrees of freedom, such as the screw joint. On the other hand, low-level constraints represent joints with independent degrees of freedom, such as a prismatic and spherical joint.

The Layout Design Tool, the PRODES system and the GeoNode system use references when connecting components. This is very useful, as it supplies the user with information on the location of the connection on the component, even though this connection has not been created. Because of this, references will be used to relate components in the conceptual design view developed here too.

In the Layout Design Tool, connections between the design spaces can be accomplished by high-level constraints between the interface features. These constraints represent the freedom between the design spaces. The conceptual design view developed here, will provide high-level modelling entities to specify the freedom between the components too. These so-called interfaces incorporate constraints that specify the freedom between the components.

All four systems described above adhere to the least commitment principle: allowing the user to omit information that is not yet known (GEMCON, PRODES), or specifying coarse geometry that can be refined in subsequent steps (GEMCON, PRODES, LDT, GeoNode). Furthermore, the GEMCON and GeoNode systems also support the specification of both vague and precise information by using linear or non-linear inequalities and equalities. Since this is a very useful principle, because it allows one to specify what is desired only, the conceptual design view developed here will adhere to the least commitment principle too.

In some of the described systems, geometry is described in a rather awkward way. In the GeoNode system, a hierarchy of local coordinate systems has to be defined before any shape can be placed. In the GEMCON system, objects can only be rotated with an angle of multiples of 90 degrees, and dimensions of different objects cannot be related. In the Layout Design Tool, the geometry of a design space can sometimes be created from a library of frequently used general forms, but otherwise needs to be defined from separate geometric elements such as faces and constraints between them. The conceptual design view developed here, will provide a simple CSG-based modelling system with attaches to allow the user to more easily create components.

In the PRODES system, the geometry of the contact surfaces is specified by surface patches. The use of surface patches enables the designer to distinguish between the regions of the geometry of a product that are functionally impor-
tant and the rest of the geometry, and is therefore profitable in conceptual design. However, because the SPIFF system does not support surface patches, these will not be available in the conceptual design view developed here. Instead, volumetric shapes will be used.

In the GEMCON system, the user has to specify all operations by typing commands like:

\[
\text{(make-geom :size'(((width = 16 \text{ -- } 18)) \text{((depth }^ = 18)) \text{((height }^ = 14)))}}
\]

The GeoNode and PRODES systems, on the other hand, provides a graphical user interface allowing the user to easily execute operations using menus. An intuitive graphical user interface is highly desirable, as it lowers the level of knowledge about the system the user has to have in order to use it. The conceptual design view developed here, will provide a graphical user interface that allows the user to intuitively specify operations without having to type complex commands.

4.2 Model elements

The feature model of the conceptual design view is built from conceptual components, shape concepts, interfaces and references.

The feature model can contain several configurations, i.e. sets of conceptual components that are related to each other, but also several unrelated components.

Conceptual components

Conceptual components are the functional units of the conceptual feature model. A conceptual component consists of a base shape with shape concepts and references specified on it. Furthermore, a conceptual component has a number of physical attributes and a function description. The base shape, the physical attributes and the function description will be described in this subsection, the shape concepts and references in subsequent subsections.

The base shape represents the global shape of the conceptual component and is used as a kind of outline to position shape concepts and references on. The geometry of a base shape should be kept simple, in order to avoid needlessly hindering the designer in the remaining process. On the other hand, it should be sophisticated enough to roughly resemble its detailed counterpart, although this is not mandatory: if the user wants to represent, for instance, an L-shaped detailed part by a conceptual component with a box-shaped base shape, he or she has complete freedom to do so. If all conceptual components resemble their detailed counterpart, then the final configuration will look like the product being designed. An example of the base shape of a conceptual component for the base grip component of the bench vice of Figure 4.7 is given in Figure 4.8.
The base shape is built from block, trapezoidal block, cylinder and truncated cone shapes, which can either add or remove material. Except the first shape, a new shape is attached to a face of a shape already in the base shape. Note that this only partially fixes the new shape with respect to the rest of the model. Additional constraints may be used to further fix the shape with respect to the rest of the model.

*Physical attributes* represent physical properties of the conceptual component, such as volume, weight and material type. In many situations, the value of such attributes is restricted by physical limitations on the component, such as the restriction that the weight should be less than 10 kg to avoid the product to become too heavy.

Four types of restrictions are available to the user, in order to restrict the value of a physical attribute. The less than and greater than restrictions specify the value of a physical attribute to be less respectively greater than a specified value. The interval restriction specifies the value of a physical attribute to be within a specified range. The equal restriction specifies an attribute to be equal to a specified value.
The function description allows the user to describe a component; what it is, where it is used for, and any other related information useful during the design process that cannot be specified anywhere else. There is no limitation on format and layout of the description, so the user is free to include any information. An example of a function description for the fixed grip component of Figure 4.8 of a bench vice would be something like:

*Fixed grip of the bench vice. Serves as the rigid part of the bench vice standing on or connected to a surface. Furthermore, it is one of the two bench vice grips.*

**Shape concepts**

A shape concept is a model element that can be used by the designer to specify functional requirements on the shape of the component. They can be included in order to guarantee that the resulting product satisfies these requirements.

A shape concept is defined on a component, and specifies a depression or a protrusion that is required for the function of the component. In addition, for each shape concept the way it is attached to the component and the type of shape, i.e. block, trapezoidal block, cylinder or cone, may be included or may be specified to be unknown. If the type of shape has been included, constraints can be specified on the dimensions of the shape. Just like for the physical attributes of a conceptual component, less than, greater than, interval and equal constraints can be used for these dimensions.

Classes of shape concepts have been defined, based on the nature and the way the shape concept is attached to the component. A special class “unknown” is available for situations in which its nature and/or the way it is attached is not known. Shape concept classes are specified in XML (Holzner 2001), which provides a structured way to specify the properties of a class.

Examples of instances of shape concept classes are shown in Figure 4.9. In this figure, the shape type and size have been specified in all cases, but these may also be left “unknown”. See Section 4.4 on how shape concepts are visualised in the latter cases.

![Figure 4.9: Instances of shape concept classes](image)

(a) (b) (c) (d) (e)

Figure 4.9: Instances of shape concept classes: isolated protrusion with cone shape (a), aligned protrusion with trapezoidal block shape (b), isolated depression with cylinder shape (c), open depression with block shape (d), and passage with block shape (e).
References

References (see Section 3.3) are used to position other model elements, such as shape concepts, and to relate interfaces to a component.

An extendible library of reference classes is available in the conceptual design view. New reference classes can be added, existing classes can be changed, and obsolete classes can be removed, just as described in Section 3.3.

Interfaces

An interface represents a reduction of freedom between the conceptual components on which it has been defined. It uses constraints to represent the reduction of freedom, but does not represent the way this reduction is enforced. Several classes of interfaces exist, such as the prismatic slide and spherical interface.

The interface classes are stored in an extendible feature library. New interface classes can be added, existing classes can be changed, and obsolete classes can be removed. New interface classes are specified in two phases.

In the first phase of creating an interface class, the number of components to be related and the classes of the references on these components, have to be specified. For example, for a prismatic slide interface, a reference with a line and an arrow could be used on each component; the line to indicate the sliding direction and to be able to partially fix the orientation of the components with respect to each other, and the arrow to be able to further fix the orientation of the components.

In the second phase of creating an interface class, the constraints between the references that represent the freedom that is reduced between the conceptual components, have to be specified. For example, the prismatic slide interface has a parallel constraint to fix the orientation of the components with respect to each other, and a co-linear constraint to specify that the components may slide with respect to each other along the line of one of the references.

Just like the form feature classes, each interface class also has an explicit class interface because it is specified in this way.

An instance of a class can be created by specifying instances of the requested reference classes on the components to be related. After that, the system positions the related components in such a way that the constraints in the instance are satisfied.

Configurations

A configuration consists of a set of conceptual components that are related by interfaces. Configurations are managed by the system, i.e. the system generates a new configuration if two unrelated conceptual components are related
by an interface, it adds a new component to a configuration if this component is related to a component in the configuration using an interface, it merges configurations if some of their components are related by an interface, it splits a configuration if unrelated subsets of components occur, and it removes a configuration if only one component is left.

### 4.3 Model representation

The feature model of the conceptual design view is represented by configuration graphs, which represent all elements that are important for the function of a product.

A *configuration graph* contains information on the way components are related by interfaces, and information on the shape concepts on the components. All these elements are represented by nodes, and relations between them are represented by edges between the nodes. Edges between a node that represents an interface and nodes that represent a component indicate an interface between components; an edge between a node that represents a shape concept and a node that represents a component indicates that the shape concept has been specified on the component. As an example, the configuration graph for the bench vice of Figure 4.7 is given in Figure 4.10.

![Figure 4.10: A possible configuration graph for the bench vice of Figure 4.7](image)

### 4.4 Model visualisation

The feature model of the conceptual design view can be visualised using graph cameras, geometry cameras, and table cameras.

*Graph cameras* visualise the structure of a configuration, i.e. the components and the interfaces between them. They use nodes containing an icon to represent both the components and the interfaces between the components, and show how these are related by edges between the nodes. An example of the graph camera for the bench vice of Figure 4.7 is given in Figure 4.11.
Geometry cameras visualise a component with its base shape, shape concepts and references, either separately or together with other components in a configuration. They visualise a base shape by means of its geometry, a shape concept by means of an icon, or its geometry if the type and size of the shape have been specified, and a reference by means of an icon.

The icon for a shape concept is described in its class, where it has been specified from lines, circles and arrows by the designer of the class. It shows the required nature, the way the shape concept is attached, and whether the shape type has been specified and, if so, what the required shape type is. The nature can be shown by an arrow that either points into the base shape to represent a depression (see Figure 4.12(a,b)) or out of the base shape to represent a protrusion (see Figure 4.12(c,d)) or, in case of a passage, a line between the entrance faces (see Figure 4.12(e,f)). The way it is attached is shown by having such an arrow for each attachment (see Figure 4.12(g,h)). A line may replace two parallel attachments (see Figure 4.12(i)). The shape type is shown by a 2D projection of the shape. For a concept with block shape, a rectangle is shown (see Figure 4.12(d,e,j,k)). For a concept with trapezoidal block shape, a trapezium or a rectangle with additional lines is shown (see Figure 4.12(c,l)). For a concept with cylinder shape, a circle is shown (see Figure 4.12(b)). For a concept with cone shape, two circles are shown (see Figure 4.12(f)). Finally, for a concept with unknown shape type, a cross is shown (see Figure 4.12(a)). The 2D projection of the shape may replace one or more arrows that show the nature and/or the way it is attached (see Figure 4.12(j,k,l)). The camera cannot visualise shape concept instances of the special class “unknown”, since these shape concepts do not contain enough information to orient and position the icon that represents the concept.
The icon for a reference is automatically generated. For an elementary reference, it depends on the type of the reference, for a compound reference it is composed of the icons of the elementary references in it.

An example of a conceptual geometry camera that visualises the conceptual fixed grip component of the bench vice of Figure 4.7, will be given here. The camera shows the base shape of the component, an icon for the protrusion shape concept with block shape that functions as the yaw of the fixed grip, an icon for the passage shape concept with block shape that functions to connect the moving grip to the fixed grip, and an icon for the prismatic slide reference that is used for the prismatic slide interface between the fixed grip and the moving grip. The camera window of this conceptual geometry camera is given in Figure 4.13.

A *table camera* shows a table with all the shape concept instances present on a component, their class and their parameters, allowing the user to have an overview of all shape concepts, even the ones that do not allow a graphical representation.
By interacting with a camera through its menu bar, the user can reuse the camera to show the shape concepts of another component and close the camera. Figure 4.14 shows an example of a table camera.

![Table Camera Window](image)

Figure 4.14: A table camera window for the shape concepts on the component of Figure 4.13.

### 4.5 Model specification

The feature model of the conceptual design view is specified by creating new components with shape concepts and references on them, and interfaces between them, and by optionally editing and removing existing components, shape concepts, references and interfaces.

#### Conceptual component specification

A conceptual component can be created by specifying a name and creating a base shape from shapes. In addition, the value of physical attributes may be restricted, and a description of the function may be given.

A simple modelling system is used to create the base shape. It provides a modelling panel to add new shapes and geometric constraints to the model, and to edit or remove existing shapes and constraints in the model. For convenience, it initially places a new shape near the centre of the shape face to which it has been attached.

An example of adding a shape to the base shape is given here. In this example, the top block for the base shape of Figure 4.8 is added and positioned. The shape is added by specifying its nature (Figure 4.15(a)), its shape (Figure 4.15(b)), the way it is attached to the model (Figure 4.15(c)) and its dimensions (Figure 4.15(d)), resulting in the model of Figure 4.15(e). Since the new shape is not yet in the desired position, two geometric distance constraints are used to further position it, as shown in Figure 4.15(f+g).
Figure 4.15: The top block is added to the base shape of the component of Figure 4.8.

An example of restricting the values of some physical attributes of a component is given here. In this example, the material and weight of a component are restricted. A restriction can be specified by selecting the check button in front of the attribute, and specifying the requested value or range of values, as shown in Figure 4.16.

Figure 4.16: The material of the fixed grip component of the bench vice should be iron, and the weight should be less than 15 kg.

**Shape concept specification**

A shape concept can be created on a component by specifying its class and by optionally specifying its shape type and constraints on the dimensions of the shape.

The shape type can be specified to be block, trapezoidal block, cylinder, cone or, in case the type of shape is not relevant, unknown. Constraints to restrict the dimensions of the shape concept may be included in a similar way
Conceptual design view

as the constraints on the physical attributes of a component, which has been shown in Figure 4.16. An example of a conceptual component with an isolated protrusion and a passage shape concept with block shape is shown in Figure 4.13.

Reference specification

A reference can be created on a component by specifying its position and orientation with respect to the base shape, shape concepts, or other references on the component.

An example of the creation of a reference will be given together with an example of the creation of an interface in the next subsection.

Interface specification

An interface of a particular type can be created between references on different components. An example of specifying an interface between the fixed grip component and the moving grip component of the bench vice of Figure 4.7, will be given here.

For the fixed grip component and the moving grip component of a bench vice to function well, they should only have a single translational freedom that allows the yaws to open and close. A prismatic slide interface can be used to specify this kind of freedom. Such an interface has to be specified between two prismatic slide references. The prismatic slide references have to be placed on the components in such a way that these components are correctly aligned if the lines of the references are co-linear and the arrows of the references are parallel. Two components with prismatic slide references that satisfy these requirements are shown in Figure 4.17(a).

Figure 4.17: Two components of the bench vice with their prismatic slide references (a) and the resulting configuration (b).
After the instance of the prismatic slide interface has been created, the system puts the components into a configuration and places them in such a way that the components satisfy the requirements represented by the interface (see Figure 4.17(b)).

4.6 Conclusions

This chapter has described the conceptual design view for the multiple-view feature modelling system. The conceptual design view supports the design of the configuration of a product, i.e. the way the product is built from components with shape concepts and references on them, and interfaces between them. It provides graph cameras to visualise the structure of a configuration, geometry cameras to visualise the base shape of a component together with the shape concepts on it, and table cameras to give an overview of all shape concepts on a component together with their attributes.

The described conceptual design view adheres to the least commitment principle: only an outline of the geometry of a component has to be specified, restrictions may, but need not, be specified on the attributes of components and shape concepts, the shape type of a shape concept may be specified unknown, and so on. These possibilities allow the designer to specify only the most important functional aspects of a product in its conceptual design view. This is important in subsequent product development phases, because it allows the designer in these phases to distinguish aspects of the model that are already fixed and aspects that can still be changed.
Chapter 5

Assembly design view

The assembly design view supports the design of connections between components. For each connection, the geometry for the connection on the components and the type of connection are specified. The relations between the assembly design view and the conceptual design view, the part detail design views and the part manufacturing planning views are described in Chapter 6.

Section 5.1 describes other approaches to feature-based assembly design. Section 5.2 presents the elements of the feature model of the assembly design view. Section 5.3 describes the way the structure and the geometry of the feature model is represented. Section 5.4 presents the different cameras that are used to visualise the feature model of the assembly design view. Section 5.5 describes the way the elements of the feature model can be specified. Finally, Section 5.6 gives some conclusions.

5.1 Other approaches to feature-based assembly design

This section will first present three existing approaches to feature-based assembly design, and will discuss them afterwards. This discussion will result in some requirements on the feature-based assembly design view developed here.

Deneux (1999) presents an approach that supports the creation and management of complex assemblies, to overcome the problem of most current commercial modelling systems that they offer only limited facilities to represent assembly information. He states that, although some of the commercial systems support dynamic assemblies to be able to do kinematic model analysis, most systems consider static assemblies only from an administrative point of view, e.g. to create bills of material.

Assembly features are defined as a generic solution to a design problem of relating components, and are created in five steps. First, two components have to be specified. After that, the design problem has to be specified, e.g. that the two components have to be fixed with respect to each other. Then, a candidate solution to solve the problem has to be selected, e.g. fix them with-
out additional components. After that, an applicable technology has to be selected, e.g. glue them together. Finally, the geometry that is needed on the components is determined, e.g. two corresponding clean surfaces, one on each component.

Cugini (2000) presents a feature-based design approach to support top-down design of assemblies used for aeronautics, leaving the design of sub-systems and details to a following phase. An assembly feature expresses a relationship that exists between two or more parts within an assembly. Assembly features are created in a way similar to that described by Deneux (1999); parts have to be chosen, a problem to be solved has to be identified, a principle solution has to be selected, and, finally, a technological solution has to be chosen.

A 3D visualisation of assemblies has been developed to efficiently interact with, and browse the structure of, large assemblies consisting of hundreds of components. The visualisation allows the designer to analyse the structure of an assembly from different points of view, and to see the various levels of detail for the assembly. For example, Figure 5.1(a) shows an attach relation between a stringer and a frame part, and Figure 5.1(b) shows a lower level with more details, in which the relation has been implemented using an assembly part, or agent, between the stringer and the frame.

![Figure 5.1: A 3D visualisation of an assembly feature at a high level of abstraction (a) and the details of its implementation at a lower level of abstraction (b) (after Cugini (2000)).](image)

Van Holland and Bronsvoort (2000) show that the feature concept is useful in both assembly modelling and planning. They introduce an object-oriented product model that supports modelling and planning of both single parts and assemblies. A single part is built from form features and constraints. An as-
Assembly is built from, among others, single parts and connection features between them. To visualise the models, geometry viewers can be used to show the geometry of the model with a line drawing or a shaded image, and graph viewers can be used to show the structure of the model with a graph (see Figure 5.2).

![Figure 5.2: The connection features between components, and the hierarchy of compound components and subcomponents in a graph viewer (after van Holland (1997)).](image)

Connection features incorporate characteristics of connection types, which can be properties when the connection has been established, but also information on how the connection can be established. Examples of assembly information for a specific connection that can be specified are involved form feature types, e.g. a rib and a slot form feature for a rib-slot connection, final position of the connected components with respect to each other, and the internal freedom of motion, i.e. the relative freedom of the components after they have been connected. Examples of connection features are PlaneMate, RibSlot and SnapFit.

In this approach, assemblies can be built using either related-driven or relation driven modelling. In related-driven modelling, the geometry of the single parts is specified first, and the relations are specified between them afterwards. In relation-driven modelling, first a relation is specified, and then
the related parts are specified. The advantage of relation-driven modelling is that it can support top-down modelling if the system would allow undetailed single parts, to which details are added during the specification of connection features.

The approaches described by Deneux (1999) and Cugini (2000) suggest that assembly features, i.e. connection features, should be specified in multiple steps, each of which adds more detail. This is a very useful approach that, for example, allows alternative implementations of connections to be compared. Since in our multiple-view feature modelling approach, the conceptual design view already takes into account connections between components at a higher level of abstraction, the assembly design view needs to provide only one level of detail for the connections to support this facility.

Van Holland and Bronsvoort (2000) and Cugini (2000) propose graphical visualisation of assemblies to efficiently interact with, and browse the structure of, assemblies. Van Holland and Bronsvoort (2000) uses 2D graphs to visualise the hierarchy of components in an assembly and the relations between them. Cugini (2000) even uses 3D graphs to visualise this information. Although 3D graphs can be very useful to visualise tree structures such as the hierarchy in an assembly (Robertson et al. 1991), they become confusing if additional connection relations are added between the nodes of the tree, see Figure 5.1(b). Therefore, the multiple-view feature modelling system will use only 2D graphs to visualise assemblies.

5.2 Model elements

The feature model of the assembly design view is built from components and connection features, which can be combined into assemblies. It can represent multiple assemblies and components.

Components

Components are the elements of the assembly feature model that are combined, and on which assembly information is defined. Components are centred around a reference frame, i.e. all elements of a component are (in)directly specified with respect to this frame. They have the form features of the connection features between them attached to them, and no other form features.

A component is either a single component or a compound component. A single component represents a part in the assembly model. A compound component encapsulates an assembly for further assembly modelling purposes, by hiding the structure of components and connection features, and dealing only with the boundary of the assembly.
As stated above, a component contains the form features of the connection features on the component. In general, the geometry of the form features of several connection features on a component is disconnected, and therefore it is difficult to comprehend the geometry of the feature model of the assembly design view. To give a better insight into the geometry, the reference geometry, i.e. the geometry of the part or assembly related to that component, is added. The reference geometry is visualised without shading to emphasise that it is not part of the feature model of the component, as shown in Figure 5.3.

![Figure 5.3: The reference geometry of a single component is based on the geometry of the related part (a) and is visualised without shading (b).](image)

The reference geometry is also used to enable a more intuitive way of creating form features of connection features on the component. These form features need not to be specified with respect to the reference frame of the component or other form features, but can also be specified with respect to the faces of the reference geometry.

The reference geometry of a component is derived from the geometry of the associated part or assembly. It needs to be regenerated whenever the geometry of the part or assembly changes. Every face of the reference geometry is assigned an unique id, to allow the features on the component to refer to them. When an existing reference geometry is replaced by a new one, each parameter of a feature that refers a face of the old reference geometry has to be updated to refer to a face of the new one.

To avoid the need for a persistent naming scheme (Chen and Hoffmann 1995; Kripac 1995) for the faces of the reference geometry, the system creates a persistent substitute for each face of the reference geometry that is selected by the user for a parameter of a form feature. It uses the substitute as the value of the parameter in the model, as is shown in Figure 5.4, and to the user, it provides the coinciding face of the current reference geometry as the value of the parameter. The persistent substitute is positioned with respect to the reference frame of the component, using geometric fix constraints, such that it coincides with the face of the reference geometry. This will be elaborated in Section 6.3.
Connection features

Connection features contain assembly information, such as the internal freedom of the connection, the types of the form features needed for the connection on the components, and the final relative position of these form features on the components. Connection features use constraints to specify this assembly information. Many different classes of connection features are possible, such as the dove-tail and the pin-hole class (van Holland and Bronsvoort 2000).

Just like the form feature classes in the part-oriented views, the connection feature classes are stored in an extendible feature library. New connection feature classes can be added, existing classes can be changed, and obsolete classes can be removed. New connection feature classes are specified in three phases.

In the first phase, the form feature classes needed for the connection have to be specified. If no suitable form feature class is available, a new form feature class has to be specified (see Section 3.3).

In the second phase, the constraints between the form features that reduce the freedom of the connected components have to be specified. These constraints are specified between elements of the form features. Examples of constraints for a dove-tail connection are geometric co-planar constraints between the corresponding side faces of the dove-tail rib and slot.

In the third phase, optional attributes for the connection feature may be specified. An example of such an attribute for a dove-tail connection is an offset to specify the relative position of the components after the connection has been established.
As a consequence of this way of specifying connection feature classes, each connection feature class has an explicit feature class interface, i.e. a description of the way the designer can interact with an instance of the connection feature class. This describes, among others, the number of components, the form features on these components, and values for the optional attributes, which have to be specified to create an instance of the connection feature class.

**Assembly**

An assembly consists of a set of connected components. Assemblies are managed by the system, i.e. the system generates a new assembly when two unconnected components are connected, it adds a component to an assembly when an unconnected component is connected to a component in the assembly, it merges assemblies when components from the assemblies are connected, it splits assemblies when disconnected subsets of components occur in it, such that each subset forms its own assembly, and it removes an assembly when only one component is left in the assembly.

**5.3 Model representation**

The feature model of the assembly design view is represented by assembly graphs that represent the model structure of the assemblies, and cellular models that represent the geometry of the components and the assemblies.

An assembly graph contains information on the way components are connected by connection features, and subcomponents are combined into compound components. Both components and connection features are represented by nodes, and the relations between them are represented by edges between the nodes. Edges between nodes that represent components indicate a compound component built from multiple subcomponents, and are directed from the compound to the subcomponents; edges between a node that represents a component and a node that represents a connection feature indicate a connection feature on the component and are undirected. As an example, the assembly graph of the bench-vice assembly of Figure 5.5 is given in Figure 5.6.

![Figure 5.5: A bench-vice assembly.](image)
The evaluated geometry of each assembly and component is represented in a cellular model (see Section 3.3). A single component shares its cellular model with the associated part, a compound component with the encapsulated assembly.

The cellular model of a single component represents the evaluated geometry of the form features of the connection features on the component, the reference geometry for the component, and the evaluated geometry of the views on the associated part. As an example, the cellular model of the grip component of the bench-vice assembly of Figure 5.5 is given in Figure 5.7(a). The integration of the evaluated geometry of the single component and its associated part in a single cellular model supports the maintenance of the consistency between a single component and its associated part, as described in Section 6.3.

Figure 5.6: The assembly graph of the bench-vice assembly of Figure 5.5.

Figure 5.7: A component represented by its own cellular model (a) and by the cellular model of the assembly it belongs to (b).
The cellular model of an assembly represents the evaluated geometry of the form features of the connection features on the components in the assembly, and the reference geometries of the components. The geometry of the form features on a component and its reference geometry is here the same as the geometry that is stored in the cellular model of the component, except that in the cellular model of the assembly, they are placed such that the connections between the components are satisfied. As an example, the cellular model of the bench-vice assembly of Figure 5.5 is given in Figure 5.7(b).

The cellular model of a compound component is shared with the encapsulated assembly. Combining the evaluated geometry of the compound component and its encapsulated assembly allows changes in the geometry of the compound component to be mapped to changes in the subcomponents, and vice versa, which will be illustrated in Section 5.5.

5.4 Model visualisation

The feature model of the assembly design view can be visualised using geometry and graph cameras.

Geometry cameras are used to visualise the geometry of components and assemblies. A geometry camera on a component visualises the reference geometry and the geometry of the form features of the connection features on the component, see Figure 5.8(a). A geometry camera on an assembly visualises the reference geometries of the components in the assembly and the geometry of the form features of the connection features in the assembly, see Figure 5.8(b).

Figure 5.8: A geometry camera on a component (a) and on an assembly (b).
Two types of graph cameras exist to visualise the structure of assemblies: hierarchical and relational graph cameras.

Hierarchical graph cameras visualise the hierarchy of an assembly with its components (see Figure 5.9(a)), and possibly subcomponents of compound components (see Figure 5.9(b)). They use nodes containing an icon to represent components, and represent the hierarchy between compound components and their subcomponents by edges between their nodes.

Nodes representing compound components in the hierarchical graph camera can be expanded and collapsed to give a better insight into the model. Expanding a compound component allows investigation of the subcomponents. Collapsing an expanded compound component recursively hides all subcomponents. Figure 5.9(b) shows a hierarchical graph of the same assembly as Figure 5.9(a), except that the node representing the compound component has been expanded.

Relational graph cameras visualise the connections between the components (see Figure 5.10). They use nodes containing an icon both to represent the components and the connection features between the components, and show how these are related by edges between them.

Nodes representing compound components in the relational graph camera can also be expanded to give a better insight into the model. When the
node that represents a compound component is expanded, it is replaced by the nodes for its subcomponents; compare Figure 5.10(a) with Figure 5.10(b) in which the bottom node has been expanded. Nodes representing subcomponents in the relational graph camera can also be collapsed. When the node for a subcomponent is collapsed, it is replaced by the node of its compound component, also hiding the other subcomponents of the compound.
When the node that represents a compound component is expanded, the nodes that were connected to that node, i.e. the nodes that represent the connection features on the compound component, have to be connected again. They are connected to the nodes that represent the subcomponents that contain the geometry of the form features for the connections on the compound. This shows the subcomponents of the compound that are effectively used to connect the compound to the other components in the model. In the example of Figure 5.10(b), the nodes that represent the connections on the fixed grip compound component are connected to the node that represents the base subcomponent, because the raised-sliding and nut-bold connections all work on the base subcomponent.

Graph cameras allow the designer to obtain a global overview of an assembly at one moment, and to do a detail inspection of some aspect of the assembly at the next moment, even when a large assembly is built from single components only. This is supported by allowing the designer to zoom out to obtain a global overview, or to zoom in to inspect some details. An example of this is shown in Figure 5.11.

![Figure 5.11: A relational graph camera showing a complete assembly (a) and one zoomed in on a particular detail (b).](image)

The icons for the nodes that represent components are generated by a geometric camera on the component with user-defined, component-dependent, camera settings. After the user has specified the camera settings for a component once, the system can automatically update the icons in the nodes that represent the component, after the geometry of the component has been changed.

### 5.5 Model specification

The feature model of the assembly design view is specified by creating components, and connection features between the components.
Component specification

A single component can be created by specifying a name and the part to which it is related, a compound component can be created by specifying a name and the assembly on which it is based.

Connection specification

A connection feature can be created by selecting components, specifying appropriate form features on the components, and specifying values for its attributes. A form feature may be selected from existing form features, may be created based on the reference geometry, or may be created by adjusting the geometry.

A form feature instance can be created based on the reference geometry by specifying values for the parameters of the form feature class, such that the corresponding instance coincides with the intended region of the reference geometry and its validity conditions are satisfied. For example, to create a raised-slider form feature based on the rib on top of the base of the component of Figure 5.12(a), the designer has to specify a slider such that its side faces coincide with the faces of the reference geometry that represent the rib, as shown in Figure 5.12(b).

Figure 5.12: An existing region of the reference geometry is used as a form feature for a connection feature.

If the reference geometry does not contain an appropriate region to create a form feature, as above, a new form feature has to be created that adjusts the geometry. If the form feature is specified on a single component, the change in the geometry is propagated to the associated part, as will be described in Chapter 6, otherwise it is propagated to the components in the encapsulated assembly.
An example of propagating the changes in the geometry of a compound component to the geometry of a single component in the encapsulated assembly will be given for a compound component that is similar to the moving grip single component of the bench vice. The compound component represents an assembly of a moving component with a hardened yaw component for a slightly different bench vice, as shown in Figure 5.13(a+b). In order to connect the wringe of the bench vice to this compound component using a pin-hole connection feature, a hole is created in the compound that changes its geometry, as shown in Figure 5.13(c).

Figure 5.13: An assembly (a) is encapsulated in a compound component (b) whose geometry is altered by a hole form feature (c).

To propagate the change in the geometry of the compound component to the geometry of a single component, the changed subcomponent is identified. It can be found by checking the shared cellular model of the compound component and its encapsulated assembly for the components that overlap with, or are adjacent to, the region of the geometry of the compound component that was changed, as is shown in Figure 5.14. The reference geometry of this component is updated to incorporate the change.

If the component whose reference geometry has been changed is still a compound component, the process is repeated. If it is a single component, the feature models of the views on the associated part are updated using consistency maintenance, which will be described in Chapter 6.

After a connection feature has been created, it can be changed, i.e. the values of the parameters of the form features and the values of its attributes can be changed. If these changes result in changes to the geometry of a component, then the associated subcomponents and/or parts will also be updated. This is similar to what was described for the creation of a connection feature.
5.6 Conclusions

This chapter has described the assembly design view for the multiple-view feature modelling approach. The assembly design view allows specification of connection features between components, encapsulation of assemblies into compound components, which can be subsequently used as “normal” components, and browsing of the structure of assemblies using graph cameras.

Separate cameras are used to browse the hierarchy and the connections between the elements of an assembly, in order to prevent an abundance of edges between nodes. Hierarchical cameras can be used to browse the hierarchical relations between an assembly and its components, and a compound component and its subcomponents. Relational cameras can be used to browse the connections between the components in an assembly.

The assembly design view supports both top-down and bottom-up design of assemblies, by allowing either newly created geometry (top-down) or existing geometry (bottom-up), to be used as a form feature for a connection feature. By also supporting both approaches to be used alternately, the designer is provided maximum flexibility to design an assembly.
Chapter 6

Consistency maintenance

In multiple-view feature modelling, it is necessary to keep several feature models consistent, because they all represent (part of) the same product. Each feature model results from a view that supports a specific phase of the development of the product.

6.1 Use of different views

This section will first describe how the supported views can be used, i.e. the order in which the models of these views can be specified, the model elements and relations between them that can be specified, and the priorities that exists between the views. After that, it will describe the pairs of views between which consistency maintenance has to be performed.

Order in which views can be used

The development of a product can be started in the conceptual design view.

After the conceptual design has become more or less stable, the development can be continued in the part detail design views. The set of parts that is created for a conceptual component should satisfy the requirements on the component.

After the geometry of some parts has been developed such that the form features for a connection feature can be created on it, assembly design can be started. For each part, a single component has to be created that should be consistent with the part. Further, a connection feature should be specified for each interface in the conceptual design view.

After some more geometry in the detail design view of a part has been created, the development can be continued in the part’s manufacturing planning view to create a manufacturing plan for the part.

Of course, the feature model of a “previous” view can always be adjusted after having worked on the feature model of a “next” view.
Relations between model elements from different views

Each conceptual component in the conceptual design view has to be related to one or more parts that together implement the functionality of the conceptual component. Each part implements functionality of one conceptual component at most, and is therefore related to one conceptual component at most. The parts that do not implement functionality of a conceptual component, may implement the functionality of an interface, e.g. an axis that allows two components to rotate with respect to each other. All parts have a part detail design view and a part manufacturing planning view.

Each shape concept in the conceptual design view has to be related to a form feature in either the part detail design view or the assembly design view that represents the shape concept, i.e. the shape concept itself has to be related to the form feature, and the parameters of the shape concept have to be related to the parameters of the form feature. However, form features in the part detail design view or assembly design view may exist that are not related to a shape concept in the conceptual design view. Such form features have a miscellaneous function, e.g. to reduce weight.

Each interface in the conceptual design view has to be related to a connection feature in the assembly design view that represents the interface. Connection features in the assembly design view may exists that are not related to an interface. Such connection features may be used to connect the components that together implement the functionality of a conceptual component.

Each single component in the assembly design view has to be related to a part that represents the single component, and each part can be related to a single component in the assembly design view. Parts that are not related to a single component may exist during the development process, but all parts should be related to a single component after the development has been completed, in order to make sure that the functionality of the conceptual design components is implemented by the assembly design view.

Single components in the assembly design view are related to the conceptual component whose functionality is implemented by the part that is represented by the single component. Obviously, single components that are related to parts that implement miscellaneous functions are not related to a conceptual component.

Priority of constraints from different views

In many situations, a change in the feature model of one view may conflict with the feature models of other views; in such situations, the feature models of these other views need to be changed. Sometimes, a feature model that is consistent with the changed feature model cannot be created for such other view, and the changed feature model should be changed again. In all cases, the feature model(s) of some view(s) need(s) to be changed in order to make all feature models consistent.
The feature model of a view may only be changed to overcome a conflict with a feature model of another view if the constraints with higher priority remain unchanged. In other words: constraints with lower priority from one view may be altered, replaced or removed to satisfy constraints with higher priority from another view.

In the approach described here, constraints that are specified in the conceptual design view have higher priority than constraints from all other views, constraints from the assembly design view generally have higher priority than constraints from the part-oriented views, and constraints from the part detail design view generally have higher priority than constraints from the part manufacturing planning view. However, some constraints from the part manufacturing planning view may get higher priority than constraints from the assembly design view and the part detail design view, in order to ensure that the product can be manufactured.

As a consequence, a change in the feature model of some view should always be propagated and a change in the feature model of another view may only be occasionally propagated. Figure 6.1 gives an overview of the pairs of views between which changes should always be propagated and the pairs of view between which changes may only be occasionally propagated. A more detailed description of some example pairs is given below.

Figure 6.1: Solid arrows indicate a pair of a source and a target view (see Section 3.3) between which changes should always be propagated, dashed arrows indicate a pair of a source and a target view between which changes may only be occasionally propagated.

The constraints in the feature model of the conceptual design view may never be automatically changed based on the constraints from the feature model of another view (arrows 1 and 2 of Figure 6.1), because the requirements that are represented by the constraints in the conceptual design view should be satisfied by all other views. However, the value of parameters in
the feature model of the conceptual design view may be updated based on
constraints from another view, as long as they keep satisfying the constraints
in the conceptual design view.

The feature model of the assembly design view may be automatically
changed based on the feature model of the conceptual design view (arrow
3 of Figure 6.1), because the constraints from the conceptual design view
should be satisfied by all other views. The feature model of the assembly
design view can only be automatically updated based on the feature model
of a part detail design view if no feature in the assembly design view needs
to be replaced (arrow 4 of Figure 6.1), because the detail design view does not
contain information on which aspects of the geometry should be represented
by form features in the assembly design view. Of course, the dimension of
existing form features and the reference geometry of a single component in
the assembly design view may always be automatically updated after the
feature model of the detail design view has been changed.

The feature model of a part detail design view may always be automatic-
cally changed based on the feature model of the conceptual design view or the
assembly design view (arrows 5 and 6 of Figure 6.1), because these latter views
represent explicit functional requirements that should be satisfied by all other
views. Only changes that do not influence functional requirements from the
conceptual design view may be automatically applied in the feature model of
the part detail design view based on the feature model of the manufacturing
planning view (arrow 7 of Figure 6.1), because manufacturing requirements
may normally not overrule functional requirements from the conceptual de-
sign view.

The feature model of the manufacturing planning view of a part may al-
ways be automatically changed based on the feature model of the detail de-
sign view on that part (arrow 8 of Figure 6.1), because the manufacturing plan
should always be consistent with the feature models of the other views.

Consistency maintenance for the views

Just like in several other multiple-view feature modelling approaches (see Sec-
tion 2.3), in the enhanced multiple-view feature modelling approach, consist-
tency maintenance is used to update the feature model of a view to make it
consistent with a feature model of another view. However, because of the dif-
f erent natures of the views in the enhanced multiple-view feature modelling
system, the feature model of a view cannot always be derived from the fea-
ture model of an arbitrary other view, and different techniques are needed for
different pairs of views.

To support the use of the views described above, consistency maintenance
needs to be performed for four source-target pairs only, which together re-
late all views. Consistency maintenance is performed between the conceptual
design view and the assembly design view, between the conceptual design
view and the part detail design views, between the assembly design view and
the part detail design views, and between the part detail design views and
the part manufacturing planning views, but not between the conceptual de-
sign view or the assembly design view and the part manufacturing planning
views. Each source-target pair has a corresponding edge in the graph of Fig-
ure 6.2.

![Diagram of view relationships](image)

Figure 6.2: The pairs of views for which consistency maintenance is performed.

These pairs of views are a minimal set of pairs of views to maintain the
consistency of the feature models of all views, which is important because the
number of consistency maintenance steps that needs to be performed, should
be minimised to reduce the number of consistency maintenance techniques
required and to optimise the performance.

### 6.2 Conceptual design view and other views

This section describes maintaining the consistency of the feature model of the
conceptual design view, and the feature model of the assembly design and
the part detail design views. It first presents the consistency definition for
these views, based on the relations between the model elements in the views.
Then it describes the way the relations between the model elements in the
views can be specified by means of associations, the way the consistency of
the feature models of the views can be checked, and the way the consistency
can be recovered if the feature models of some views have been found to be
inconsistent.

#### Consistency definition

The consistency definition specifies when the feature models of two views are
consistent.
The feature model of the conceptual design view is consistent with the feature models of the assembly design view and the part detail design views if the conceptual components are consistent with the related assembly components and parts, and the interfaces between the conceptual components are consistent with the related connection features between the assembly components.

A conceptual component is consistent with its related assembly components and parts, if the requirements on the volume, material and weight attributes of the conceptual component are consistent with the parts, and the shape concepts on the conceptual component are consistent with the related form features on the assembly components and parts. A shape concept is consistent with its related form feature, if its class and its shape-type attribute are consistent with the form feature, and the dimensions specified for the shape concept are consistent with the related dimensions of the form feature. The class of a shape concept is consistent with a form feature, if the form feature is attached as intended in the shape concept, and the nature of the form feature is equal to the nature intended in the shape concept. The shape-type attribute is consistent with a form feature if the shape of the form feature is equal to the specified shape type. A dimension requirement of a shape concept is consistent with a related dimension of a form feature, if the value of the dimension of the form feature satisfies the requirement.

An interface is consistent with its related connection feature if their degrees of freedom and, if applicable, the direction of their freedom, are consistent.

**Specification of relations with associations**

A designer can use different types of associations to relate model elements from the conceptual design view to model elements from the assembly design view and part detail design views, and thus to specify the consistency definition for these views.

*Component associations* can be used to relate a conceptual component to one or more assembly components or parts. If an assembly component or part has already been associated to a part or assembly component, respectively, then an additional association is automatically created between the conceptual component and that other element. If the assembly component happened to be a compound component, then additional associations are automatically created between the conceptual component and all parts that have been associated with a single component within the compound component.

The GUI that supports the creation of component associations, by providing two buttons with a popup menu to select the views and two selection lists to select an element in each view, is shown in Figure 6.3(a). This GUI also supports the removal of component associations by providing a selection list that contains all existing component associations, denoted by an icon that shows the views of the associated elements and by the names of the associated elements.
A shape concept - form feature association can be used to relate a shape concept of a conceptual component to a form feature of a related assembly component or part.

To avoid the need to manually relate the dimensions of the shape concept to the dimension of the related form feature, the system automatically relates the dimensions with the same name, and informs the designer on it. Later on, the designer is allowed to change this.

The GUI that supports the creation of shape concept - form feature associations, by providing a drop down list box to select a component association and two selection lists to select a shape concept and a form feature, is shown in Figure 6.3(b). This GUI also supports the removal of these associations.

A parameter association can be used to relate a parameter associated with a shape concept, e.g. a dimension, to a parameter of a related form feature. Parameter associations are created automatically between the parameters of the elements of shape concept - form feature associations that have equal name,
when a shape concept is associated to a form feature. The designer may refine the set of automatically generated parameter associations, by removing some of them and adding others.

The GUI that supports the creation of parameter associations, by providing drop down list boxes to select a component and a shape concept - form feature association, and two selection lists to select a parameter of a shape concept and a parameter of an associated form feature, is shown in Figure 6.4(a). This GUI also supports the removal of parameter associations.

![Figure 6.4: GUIs to specify parameter associations (a) and interface - connection feature association (b).](image)

An interface - connection feature association can be specified to relate an interface to a connection feature.

The GUI that supports the creation of interface - connection feature associations, by providing a selection list to select an interface and a selection list to select a connection feature to be associated, is shown in Figure 6.4(b). This GUI also supports the removal of interface associations.
Consistency checking

The consistency of the feature models is checked by the system after each modelling operation.

The feature models with the associations between them are said to be consistent, if they satisfy the consistency conditions presented before in the subsection on consistency definition. For each type of association, a scheme has been developed to check the consistency conditions involved.

For each component association that relates a conceptual component with one or more parts, it is checked whether the requirements on the volume, material and weight attributes of the conceptual component are consistent with the parts.

A constraint solving scheme is used to check this. Each part is provided with three algebraic variables, and algebraic constraints on them that specify the values of two of them to be equal to the volume and weight of the part, respectively, and the value of the third of them such that it represents the material. The algebraic constraint solver is used to check whether the algebraic constraints that represent the requirements in the conceptual design view are consistent with these constraints in the part detail design views. In case a conceptual component is related to a single part, the solver finds them consistent if they can be satisfied together with algebraic equal constraints between the corresponding attributes. In case a conceptual component is related to more than one single part, a plus constraint is used between the volume and weight attributes of the conceptual component and the related parts instead, stating that the volume and weight of the conceptual component should be equal to the sum of the volumes and weights of the parts.

For each shape concept - form feature association, it is checked whether the class and the requirements on the shape-type attribute of the shape concept are consistent with the form feature.

To be able to check this, each form feature and shape concept is provided with three algebraic variables and algebraic constraints on them that specify the values of the variables such that they represent the way the form feature or shape concept is attached, its nature and its shape type. Again, the algebraic constraint solver is used to check whether these algebraic constraints can be satisfied together with algebraic equal constraints between the variables of the form feature and the shape concept that represent the same aspect.

For each parameter association, it is checked whether the algebraic constraints on the parameter of the shape concept are consistent with the value of the parameter of the form feature.

Again, the algebraic constraint solver is used to check this.
For each interface - connection feature association, it is checked whether the degrees of freedom and, if applicable, the direction of freedom are consistent with the freedom represented by the connection feature.

To be able to check the consistency of the degrees of freedom and the direction of freedom, both the interface and the connection feature are provided with algebraic variables that represent the number of degrees of freedom and geometric variables that represent the directions of the degrees of freedom.

The algebraic constraint solver is used to check whether the number of degrees of freedom is consistent, and the geometric constraint solver is used to check whether the directions of the degrees of freedom are consistent. In this way, situations with independent translational and rotational degrees of freedom can be dealt with.

**Consistency recovery**

If consistency checking finds an inconsistency between the feature models of two views, one of the feature models has to be changed to make the feature models consistent again.

The system provides an inconsistency browser window with tips on the aspects of the feature models that have become inconsistent to inform the designer, who is supposed to make the feature models consistent again. After each change to a model, the consistency of the models is checked again, and, if still necessary, the tips are updated.

The contents of a tip is based on the association that caused an inconsistency to be found and the view of the designer. It specifies the elements of the association and, if applicable, the involved attributes of these elements. The tip is expressed in terms of the view of the designer, because usually a tip is more comprehensible if it is in terms of that view.

A tip directly refers to elements in the view of the designer, and refers to elements from other views as related elements. For example, in case an inconsistency occurs between a requirement on the volume of a conceptual component and the actual volume of its related part, the tip for the designer in the conceptual design view would be that a requirement on the volume of the conceptual component is not satisfied by the related part, whereas the tip for the designer in the part detail design view would be that the volume of the part does not satisfy a requirement on the volume of the related conceptual component.

**Examples**

Three examples of tips to recover from an inconsistency between the feature models of two views will be given here.
In the first example, the moving grip component of the bench vice in the conceptual design view (see Figure 4.7) has been related to a part that implements the moving grip component (see Figure 6.5(a)), using a component association. In the conceptual design view, the material of the component has been specified to be cast iron, and its weight has been restricted to be less than 20 kg, to avoid that the vice becomes too heavy. At the moment the component has been related to the part, the part satisfies these requirements.

![Figure 6.5: The detail design of the part that implements the moving grip component of a bench vice (a), and an alternative (b).](image)

Now, additional requirements on the strength of the moving grip component require the detail design of the part to be changed; it is made more robust, as shown in Figure 6.5(b). However, this causes the volume of the part to increase such that it would weigh 22 kg, if made of cast iron. Since this is inconsistent with the requirement on the weight of the related component, the detail design view on the part and the conceptual design view on the component have become inconsistent.

After the operation, the consistency checking algorithm detects the inconsistency and the system provides a tip to inform the designer, as shown in Figure 6.6. The tip is based on the association that caused the inconsistency to be found and the view of the designer, and specifies the elements of the association, and if applicable, the involved attributes of these elements. Subsequently, the designer can change the part in the detail design view or the requirements on the component in the conceptual design view, in order to make them consistent again. The tip will disappear when the designer has removed the inconsistency.

In the second example, a shape concept - form feature association is created between the isolated depression shape concept with block shape on the moving grip component in the conceptual design view, which is used to make room for the wringe and the fixed grip components, and the blind slot form feature in the detail design view on the part that is related to the component, which was intended for that purpose. This results in an inconsistency between the
Figure 6.6: The tip that is provided to inform the designer on the inconsistency between the moving grip component of the bench vice and its part.

part detail design view and the conceptual design view, because the blind slot is placed in a different way than implied by the related isolated depression shape concept.

After adding the association, the system finds the inconsistency and provides a tip to inform the designer, as shown in Figure 6.7(a). The designer subsequently concludes that it is best to adjust the type of the shape concept in the conceptual design view, because it should indeed be an open depression instead of an isolated one. In order to make this adjustment, the designer switches from the part detail design view to the conceptual design view. As a result, the modelling system updates the tip to express it in elements in the conceptual design view (see Figure 6.7(b)).

In the third example, the radius of a passage shape concept with cylinder shape on the moving grip component in the conceptual design view has been related to the radius of a hole form feature on the moving grip component in the assembly design view. After that, an interval constraint is added to the radius of the passage shape concept in the conceptual design view, which specifies the radius to be between 3 and 4 cm. This causes the conceptual design view to become inconsistent with the assembly design view, because the radius of the hole feature in the assembly design view had been specified to be only 2 cm.

As a result, the system provides a tip to inform the designer on the inconsistency that occurred, and the designer is allowed to change one of the feature models in order to make them consistent again.
6.3 Assembly design view and part detail design views

The previous section already described how the feature model of the assembly design view is kept consistent with the feature model of the conceptual design view. This section will describe how the consistency between the feature model of the assembly design view and the feature models of the part detail design views is maintained.

Consistency definition

The feature model of an assembly design view is consistent with the feature models of the part detail design views, if each single assembly component and
its associated part are consistent. A single component and a part are consistent if they have the same geometry or, stated more technically, if the features on the single component are consistent with the feature model of the part detail design view, i.e. if all additive features on the single component overlap with a region of the model of the associated part with additive nature, and all subtractive features on the single component either do not overlap with the model of the associated part or overlap with a region with subtractive nature.

For example, the feature model on the part shown in Figure 6.8(a) is consistent with the feature model of the associated single component shown in Figure 6.8(b), because the nature of the overlapping regions of the models is the same, and the feature model on the part shown in Figure 6.9(a) has become inconsistent with the feature model on the associated single component shown in Figure 6.9(b), because a hole feature for a pin-hole connection has been added to the single component, which overlaps with a region of the part that has additive nature.

Figure 6.8: Consistent feature models of the detail design view on a part (a) and the assembly design view on its associated single component (b).

Figure 6.9: The feature model of the detail design view on a part (a) is inconsistent with the feature model of the associated single component (b) because a hole is missing in the detail design view.
Propagation of changes

Propagation of changes is used to try to avoid the feature model of the assembly design view from becoming inconsistent with the feature models of the part detail design views. It uses link constraints to propagate changes in the feature model of one view to the feature model of another view.

Three types of link constraints are used to propagate changes between the feature model of a single component and the feature model of its associated part in the part detail design view.

The first type of link constraints is the same as the one that is used for propagation of changes between a part detail design view and the corresponding part manufacturing planning view, and is used to link feature faces from both views that represent the same model face.

The second type of link constraints is used to link a face of a feature in the part detail design view that represents a model face, to a face of the reference geometry of its associated single component that represents the same model face. This type of link constraints is used to fix the faces of the features in the part detail design view that do not overlap with any faces of the features on the single component.

The third type of link constraints is used to link a persistent substitute (see Section 5.2), which represents a face of the reference geometry that is referred to by a model element in the assembly design view, to a face of a feature in the part detail design view that represents that same model face. This type of link constraints enables the system to update the parameters of features in the assembly design view that refer to faces of the reference geometry, when the reference geometry is changed. The link ensures that a persistent substitute remains coincident with its linked model face as long as the model face does not disappear. The parameter of the feature should be updated to refer to the face of the new reference geometry that represents that model face. If the model face that was represented by a face of the old reference geometry has been split or has disappeared from the model, the designer has to be involved to select a new face for the parameter of the feature.

An example of updating the parameters of features on the moving grip component for a bench vice similar to the one of Figure 6.9 will be given here. To facilitate the wringe that is used to open and close the vice, the height of the base of the moving grip component is increased and a blind slot is added. This is done in the detail design view on the part that is associated with the moving grip component. Original part and component are shown in Figure 6.10(a+b).

To keep the feature model of the component consistent with the part detail design view, all persistent substitutes in the assembly design view are linked to the related feature face in part detail design view using this type of link constraints. One such link is shown between the models of Figure 6.10(a) and Figure 6.10(b). This link constraint ensures that the persistent substitute is moved whenever the feature face to which it is linked moves. Here the
Figure 6.10: When the feature model of a part is updated, then a new reference geometry is generated. Replacing the old reference geometry of the associated assembly component by the new one requires the values of the feature parameters that refer to faces of the old reference geometry to be updated.

Now, the reference geometry of the component is replaced by an updated one, which is shown in Figure 6.10(e), to reflect the changes in the geometry of the associated part. Updating the reference geometry requires the value of the feature face to which it is linked is moved up to increase the height of the base, as shown in Figure 6.10(c), and, as a result, the persistent substitute and the features that referred to the face of the reference geometry that is related to the persistent substitute also move up, which is shown in Figure 6.10(d). Note that two features have now become detached from the old reference geometry, but still overlap with features in the feature model of the part detail design view.
parameters of the features that refer to faces of the reference geometry to be updated. For this, the new reference geometry is searched to find a face that coincides with the persistent substitute that was related to the referred face of the old reference geometry. If such a face is found, the value of the parameter of the relevant feature is set to the face of the new reference geometry, otherwise the designer is consulted. He then has to choose another face as value of the parameter.

The new feature model of the component is shown in Figure 6.10(f).

A special constraint management scheme is used to make sure that the link constraints can propagate changes in the feature model of one view to the feature model of another view. The scheme deactivates the dimensions of the features in the latter view, in order for them to be determined, via the link constraints, by the feature model of the view that is changed (Dohmen 1998).

Link constraints can only propagate simple changes, and can in fact only keep feature models consistent if the topology of the models does not change. Unfortunately, the topology of a feature model often changes, e.g. when a new feature is added to the model. In such situations, propagation of changes fails and consistency checking determines that the model has become inconsistent (de Kraker et al. 1997).

**Consistency checking**

The consistency of the feature model of a single component in the assembly design view and the feature model of the part detail design view on the associated part, is checked by considering the natures of the cells in their cellular model.

For each cell in the cellular model that overlaps with a feature on the single component, it is checked whether the cell either has the same nature in both views, or has subtractive nature in the assembly design view and does not overlap with the feature model of the part detail design view. If a cell is found that does not satisfy this condition, the feature models are said to be inconsistent, and consistency recovery needs to be performed to restore their consistency.

Figure 6.11(a) and Figure 6.11(b) show the cells in the cellular models of the models shown in Figure 6.8 and Figure 6.9 that are relevant to check the consistency between the parts and their associated components. These are the cells that overlap with the form features on the single components that are used for the connections between components. Because the cells of Figure 6.11(a) all have the same nature in the feature model of the part and in the feature model of the associated component, this part and component are found to be consistent. On the other hand, because the cylindrical cell in the front of Figure 6.11(b) has additive nature in the feature model of the part but subtractive nature in the feature model of the associated component, this part and component are found to be inconsistent.
Consistency maintenance

If consistency checking finds an inconsistency between the feature models of the assembly design view and a part detail design view, one of the feature models has to be adjusted to make them consistent again.

If a change to the feature model of a single component caused the feature model of its associated part to become inconsistent, then the feature model of the part can be made consistent again automatically. Feature recognition is used for this. It can create a new feature model for the detail design view on the part that is consistent with the feature model of the component in the assembly design view and the rest of the detail geometry of the part.

If a change to the feature model of a part caused the feature model of the associated single component to become inconsistent, then the feature model of the single component has to be made consistent again by the designer, although the reference geometry can be updated automatically. The feature model of a single component cannot be automatically made consistent with the feature model of its associated part, because it should only represent the regions of the feature model of the part that are involved in the connections between the components.

Example

An example of the automatic recovery from inconsistencies between the feature model of the assembly design view and the feature model of a part detail design view, will be given here.

In this example, the base grip component of the assembly design view on the bench vice of Figure 5.5 is adjusted to facilitate a nut-bolt connection feature between the base grip component and the wringe component. The model of the part detail design view on the part that is related to the component and the component itself are shown in Figure 6.12(a+b).
When the nut form feature (a kind of through hole) for the nut-bolt connection is added to the component, the component becomes inconsistent with the part detail design view on the related part; compare the model of Figure 6.12(a) with the model of Figure 6.12(c) that contains the new form feature. To make the model of the part consistent with the component again, the system uses feature recognition. The updated model for the part, which includes a new, automatically created through hole form feature, and is thus consistent with the component, is shown in Figure 6.12(d).

6.4 Conclusions

This chapter has described the way the consistency of feature models of the conceptual design view, the assembly design view and the part detail design views can be maintained. It has introduced associations to specify relations between elements from different views; these have to be specified by the designer. It has also described the way the system checks the consistency of the feature models of the views by means of these associations, and the way the consistency of the feature models is recovered when two feature models have been found to be inconsistent.

In the enhanced multiple-view feature modelling approach presented here, the associations between the elements of the feature models have to be specified manually. Existing multiple-view feature modelling approaches only deal with a single part, and form feature models that should all represent the same model geometry (see Chapter 2). In such approaches, there is no need to manually specify relations between the elements of different feature models, because such relations can be created automatically in a straightforward manner. However, these approaches are not suitable to support early product development phases, in which the geometry is only partially known, or to support development phases that deal with multiple components and the relations between them.
The feature models of the views in the enhanced multiple-view feature modelling approach can support some of these development phases, but are much more diverse as a consequence, which makes it necessary for the designers to specify the associations between them. Unfortunately, the designer should not only specify the associations once, but should replace existing ones and add new ones when major changes are made to the feature models. Finding ways to automatically generate associations between views, or even features in a target view from features in a source view with less information, would be very useful. This is related to research on automatically generating geometry based on a functional description (Gardan et al. 2001; Roy et al. 2001).

In case the consistency of the feature models of two views needs to be recovered, the enhanced multiple-view feature modelling approach sometimes has to involve the designer, because the heterogeneous character of the feature models makes it impossible to always automatically recover from an inconsistency between two feature models. If a change in the feature model of a single component in the assembly design view has caused it to become inconsistent with the feature model of its associated part in the part detail design view, then the system can often automatically recover the consistency of the feature models. Also, if a change in the feature model of the detail design view on a part made it inconsistent with the feature model of the manufacturing planning view on that part, or vice versa, then the system can often automatically recover the consistency of the feature models. In the other situations, the designer is supported to manually recover the consistency.

Although the designer is supported in consistency recovery, this support may be extended, e.g. by automatically adjusting a feature model of one view in order to make it consistent with the feature model of another view, whenever this is possible. Given the fact that in many situations the value of a parameter of a model element is not critical, the system could adjust the values of non-critical parameters to make a feature model consistent with another feature model. Of course, this is only possible if the designer has initially specified which parameters have a value that is non-critical.

For views with form feature models, automatic adjustment of a changed feature model for which a consistent feature model in another view is not possible, will be described in the following chapter.
Chapter 7

Automatic model adjustment

As has already been stated in the preceding chapters, it is very important that each feature model of a product remains valid in itself, and, in addition, consistent with the other feature models of the product at the same time. In current feature modelling systems, an invalid feature model always has to be manually adjusted in order to make it valid again, and a feature model for which a consistent feature model in another view does not exist also has to be manually adjusted in order to make such a consistent feature model possible.

The main disadvantage of manually adjusting a feature model is that it can be a lot of tedious work for a designer to get a model that satisfies all requirements on the product at the same time. A way to avoid this is to have the system automatically adjust the feature model. In this chapter, an automatic model adjustment approach for the part-oriented views is presented. To allow automatic adjustment of a model, it should contain information on which parameters are allowed to change, and which are not because they represent a hard requirement on the represented product.

In the multiple-view feature modelling system described here, the designer is allowed to indicate whether the specified value of a dimension of a form feature in a part-oriented view is critical, or whether it may be changed within a certain range. This information allows the system to automatically adjust an invalid feature model or a feature model for which no consistent feature model in the other part-oriented view is possible, such that the model becomes valid or a consistent feature model in the other view becomes possible.

Section 7.1 describes in detail the scheme that is used to determine whether a feature model is valid or not. Section 7.2 presents the concept of automatic model adjustment in the context of model validity maintenance. Section 7.3 presents the techniques used to automatically adjust invalid feature models. Section 7.4 describes the use of automatic model adjustment in consistency maintenance. Finally, Section 7.5 gives some conclusions and discussion.

7.1 Model validation scheme

Model validation determines the values of the parameters, and checks whether all constraints in the model have been satisfied. In the multiple-view feature
modelling system, model validation is based on the feature dependency graph (see Section 3.3), and uses a number of constraint solvers and constraint checkers that have been integrated in a solving scheme.

The solving scheme manages a geometric constraint solver, an algebraic constraint solver, a dimension constraint checker, and an interaction constraint checker (see Figure 7.1). The geometric and algebraic constraint solvers are used to determine the size, the position and the orientation of the feature shapes. The dimension constraint checker is used to determine whether all dimensions are within the ranges that have been specified by dimension constraints. The interaction constraint checker is used to determine whether no disallowed overlap between the feature shapes occurs.

![Figure 7.1: The solving scheme and the moments that it decides whether a model is valid or not.](image)

In order to integrate the solving of algebraic and geometric constraints, the solving scheme repeatedly applies the algebraic constraint solver and the geometric constraint solver, until none of them changes the model any more (Dohmen 1998; Hoffmann and Joan-Arinyo 1997). This allows feature models with dependencies between the algebraic and the geometric constraints to be solved, although cyclic dependencies between geometric and algebraic constraints are not allowed, because these would prevent the solving scheme from converging.

Each constraint solver and constraint checker works on its specific constraint model, which is derived from the feature dependency graph. The constraint model for a specific constraint solver or constraint checker contains only those constraints and constraint variables that are relevant for that solver or checker.
The geometric constraint solver is based on the constraint solving approach described by Kramer (Dohmen 1998; Kramer 1992), which uses degrees of freedom analysis.

The constraint model here contains a constraint variable for each face of a feature in the feature dependency graph. The constraint variables in the constraint model are called geoms and have a position and orientation. The geometric constraints from the feature dependency graph are usually mapped onto a number of low-level geometric constraints in the constraint model. For example, a co-axial constraint is mapped onto an in-line constraint that specifies a point to be on a line and a parallel constraint.

After the constraint model has been solved by the geometric constraint solver, the position and orientation values of the constraint variables that represent the faces of the features in the feature dependency graph are updated from the values in the solved constraint model.

The algebraic constraint solver is based on the SkyBlue solver (Sannella 1992), which uses local propagation.

The constraint model here contains a constraint variable for each dimension of a feature and for each user-defined variable in the feature dependency graph that are used in algebraic constraints. The algebraic constraints from the dependency graph are mapped onto one or more low-level SkyBlue constraints in the constraint model.

After the constraint model has been solved by the algebraic constraint solver, the values of the constraint variables that represent the dimensions of the feature shapes or the user-defined variables in the feature dependency graph are updated from the values in the solved constraint model.

The dimension constraint checker checks whether the dimensions of feature shapes and the user-defined variables satisfy the dimension constraints that have been specified on them. The check is performed by comparing the actual value of the variable with the allowed range, as it has been specified in the dimension constraint.

The interaction constraint checker checks whether no disallowed interaction between feature shapes occurs. The checker is based on the cellular model (see Section 3.3).

The information in the cellular model can be used to check whether the shapes of two features overlap and whether feature faces are on the boundary, and therefore to check interaction and boundary constraints (Bidarra and Bronsvoort 2000b).

Using these four solvers and checkers, the solving scheme decides whether a model is valid or not. A model is invalid if one of the constraint solvers cannot satisfy a constraint, or one of the constraint checkers finds a constraint
The cause of a feature model to be invalid is a clash of some constraints in it, i.e. some constraints prevent another constraint to be satisfied. Involving the user to make an invalid model valid again has been described by Bidarra and Bronsvoort (2000b). Their approach provides the user with corrective hints for each unsatisfied constraint. However, these hints are only indicative, in that they only indicate a feature and an operation type, i.e. edit or remove, on that feature. Corrective actions can then easily result in other unsatisfied constraints, and therefore making the model valid again manually can be rather tedious, in particular for complex models.

If one of the constraints is related to a feature parameter that is not critical, however, it will be shown that the system can mostly automatically adjust this feature parameter in order to remove the clash, and thus enforce model validity.

Mäntylä (1990) already describes the use of the strength concept of DeltaBlue (Freeman-Benson et al. 1990) to distinct three levels of commitment for a value of a variable, i.e. anchored, default and don’t care. Based on the level of commitment for each variable, the system automatically adjusts an invalid model that results from a modelling operation, in order to make it valid, if possible.

The shortcoming of this approach is that it is based on the specific capabilities of the DeltaBlue constraint solver, and this constraint solver can only deal with a limited subset of models and constraints. In particular, it cannot deal with any cyclic dependencies in the model, and it cannot be used to check interaction constraints. For this, other constraint solvers have to be used that do not, and cannot, support the strength concept.

The automatic model adjustment method that is presented here is built on top of the constraint solvers that are currently used to solve or check geometric, algebraic, dimension and interaction constraints. It can deal with clashes between constraints of the same type and constraints of different types. An overview of the invalid situations that can be resolved by the automatic model adjustment method is given in Table 7.1.
The principle of automatic model adjustment

<table>
<thead>
<tr>
<th>unsatisfied cause</th>
<th>explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>geometric geometric</td>
<td>a geometric constraint could not be satisfied because of other geometric constraints</td>
</tr>
<tr>
<td>geometric algebraic</td>
<td>a geometric constraint could not be satisfied because of the value of one of its algebraic variables that is determined by algebraic constraints</td>
</tr>
<tr>
<td>algebraic algebraic</td>
<td>an algebraic constraint could not be satisfied because of other algebraic constraints</td>
</tr>
<tr>
<td>algebraic geometric</td>
<td>an algebraic constraint could not be satisfied because of the value of a variable that is determined by geometric constraints</td>
</tr>
<tr>
<td>dimension geometric</td>
<td>a dimension is out of range because of geometric constraints that determine the value of the dimension</td>
</tr>
<tr>
<td>dimension algebraic</td>
<td>a dimension is out of range because of algebraic constraints that specify the value of the dimension</td>
</tr>
<tr>
<td>interaction geometric</td>
<td>feature shapes overlap in a disallowed way because of geometric constraints that relate them</td>
</tr>
<tr>
<td>interaction algebraic</td>
<td>feature shapes overlap in a disallowed way because of algebraic constraints on their dimensions</td>
</tr>
</tbody>
</table>

Table 7.1: Overview of the invalid situations that can be resolved by the automatic model adjustment method, denoted by the type of unsatisfied constraint and the type of the constraints that caused the constraint to be unsatisfied.

The model adjustment method only adjusts certain feature parameters. In many situations the exact value of a feature parameter is not critical, but may be changed within a given range (Ilies and Shapiro 1997). To specify this, a designer can mark a feature parameter that represents a dimension to be non-critical. Feature parameters that have been marked to be non-critical are called *variant feature parameters.*

The model adjustment method has been integrated with the solving scheme of Figure 7.1. Automatic model adjustment is started after one of the constraint solvers could not satisfy a constraint, or one of the constraint checkers found a constraint unsatisfied, i.e. after an invalid model has been found (see Figure 7.2).
It starts with finding the variant feature parameters to be adjusted. If a geometric or an algebraic constraint could not be satisfied, or a dimension constraint has been found unsatisfied, the variant feature parameter to be adjusted can be determined using dependency analysis. If an interaction constraint has been found unsatisfied, the variant feature parameter to be adjusted cannot be determined in a straightforward way, and a trial-and-error approach is used. Both approaches will be described in detail in the next section.

After that, the changes of the variant parameters that are needed to make the model valid again, need to be determined. In case the model is invalid because of a geometric, an algebraic or a dimension constraint, these changes can
be determined using constraint solving. In case the model is invalid because of an interaction constraint, a more or less randomly determined adjustment is used.

In both cases, if no appropriate variant feature parameter is present in the model, the model has to be adjusted manually.

7.3 Adjustment strategies for different types of constraints

This section describes the adjustment strategies for the different types of constraints. It first describes the adjustment strategy for geometric constraints. After that, it describes the incorporation of algebraic and dimension constraints in this strategy. Finally, it describes the adjustment strategy for interaction constraints.

Adjustment strategy for geometric constraints

The adjustment strategy for geometric constraints is based on dependency analysis of the geometric constraint model. Dependency analysis, in general, determines the dependencies between the variables in the model, e.g. that the value of variable A depends on the value of variable B and the value of variable C. Dependency analysis can be performed by the geometric constraint solver during the solving of the geometric constraints. However, dependency analysis can also be done independently of the constraint solver. The result of the dependency analysis is stored in a dependency graph. This graph is the basis for the adjustment strategy for geometric constraints.

The dependency graph consists of nodes that represent the geoms in the constraint model of the geometric constraint solver and nodes that represent the constraints from that model. The nodes that represent the constraints contain information on the degrees of freedom that are reduced by the constraint. This information can be used to find the sequence of constraints between two geoms that reduce the degrees of freedom between the geoms in a given way, i.e. it can be used to find the dependencies between the geoms.

Based on the dependency graph and information on which feature parameters are variant, an invalid feature model can be made valid again. An example of this will be given based on a model that consists of a rectangular base feature, a trapezoidal rib on top of it, and two triangular stiffeners on them; it is shown in Figure 7.3.

In this example, the user adds a model validation constraint to force the left and the right stiffener in the same position. The result of this operation is an invalid model, because the existing positioning constraints on the stiff-
Figure 7.3: A model with variant positions for the stiffeners.

eners clash with the new constraint between them, and therefore, the new
constraint cannot be satisfied. In order to be able to adjust the model to make
it valid again, the dependency graph of the model is built. It is shown in
Figure 7.4, with the shapes as geoms and the rotational freedom discarded
\((T(0,1,0))\) indicates that a constraint reduces a translational degree of freedom
in the direction \((0,1,0)^T\). Including the separate elements of the shapes and
the rotational freedom in the figure, would result in many additional nodes
that are irrelevant for this example.

Figure 7.4: The dependency graph of the model of Figure 7.3 with the model constraint
(newdist) added.

The first step of automatic model adjustment is to find the constraints in
the model that clash. In the dependency graph, these constraints can be found
by finding a loop in the graph with constraint nodes that reduce the same
freedom. In Figure 7.4, these nodes have been shaded somewhat darker.

The second step of automatic model adjustment is to determine the variant
feature parameters that are related to these constraints. In this example, the
variant position of the left stiffener is related to the dist7 constraint, and the
variant position of the right stiffener is related to the dist4 constraint. One of
these variant feature parameters is randomly selected by the system.
The final step of automatic model adjustment is to adjust this variant parameter in order to enforce model validity. For geometric constraints, the needed adjustment is determined by temporarily removing the constraint related to the chosen parameter, and solving the constraint model by the geometric constraint solver. The result of this is a new value for the variant feature parameter that does not clash with the new constraint. After solving the constraint model, the value of the parameter of the removed constraint is updated according to the new value found, and the constraint is added again.

**Incorporating algebraic and dimension constraints**

The adjustment strategies for algebraic and dimension constraints are very similar, because a dimension constraint is a special kind of algebraic constraint. It is an algebraic unequal constraint, or a pair of algebraic unequal constraints, that specifies a dimension of a feature model to be larger or smaller than a given value, or to be between two values. Therefore, only the adjustment strategy for algebraic constraints will be dealt with here.

This adjustment strategy is based on dependency analysis of the algebraic constraint model. This is easier than dependency analysis of a geometric constraint model, because the algebraic constraint variables have only one degree of freedom. Because of this, the adjustment strategy for algebraic constraints will not be described separately, but integrated with the adjustment strategy for geometric constraints.

The algebraic constraint model and the geometric constraint model are related by algebraic constraints on algebraic variables of geometric constraints, such as the distance variable of a distance-face-face constraint. Algebraic constraints clash with geometric constraints if the geometric constraints specify a different value for an algebraic variable than the algebraic constraints do. Incorporating algebraic constraints in the model adjustment scheme for geometric constraints, requires extensions of the scheme with respect to finding the constraints that clash.

The dependency graph of the algebraic constraints is integrated with the dependency graph of the geometric constraints. This is done by creating nodes that represent the algebraic variables of geometric constraints and linking these nodes to the nodes that represent the geometric constraints, and by creating nodes that represent the algebraic constraints and linking these to the nodes that represent the algebraic variables.

An example of the adjustment strategy that incorporates both algebraic and geometric constraints is based on a model similar to the model of Figure 7.3. In this model, the trapezoidal rib is attached to both the front face and the back face of the base, so the width of the rib is always equal to the width of the base, and the widths of the base and the stiffeners are variant, but the stiffeners should have a minimal width of 4 cm due to strength requirements (see Figure 7.5).
Automatic model adjustment

Figure 7.5: A model with a variant width for the base and the stiffeners.

Now, in addition to the other requirements on the product, it should be possible to add optional handle bars on the right side of the rib. To facilitate space for the handle bars, the user adds an algebraic plus constraint between the width of the rib and the width of the right stiffener that specifies the width of the rib to be 16 cm larger than the width of the stiffener. This causes the model to become invalid, because the width of the rib (15 cm) and the width of the stiffener (4 cm) do not match.

The dependency graph that integrates the algebraic and geometric constraints in this model is shown in Figure 7.6. Because the dimensions of the shapes in this model are part of the problem, the figure shows multiple nodes for the shapes. Again, in order to increase the insight into the dependency graph, only the relevant nodes are shown. Notice also that the constraint node that represents the constraint for the distance between the front and the back of the rib does not reduce any freedom, because this distance is determined via the attaches between the rib and the base, and the distance between the front and the back of the base.

The first step of automatic model adjustment is again to find the constraints in the model that clash. From the dependency graph in Figure 7.6, it can be found that the new algebraic constraint, i.e. plus1, enables the width of the right stiffener, i.e. width3, to specify the width of the rib, i.e. width2. The latter had already been implicitly specified by the width of the base block, i.e. width1, and thus it is concluded that plus1 clashes with width1 and width3. It does not clash with width2, because this constraint does not reduce any freedom.

The second step is to determine the variant feature parameters that are related to these constraints. In this example, both the width of the base and the width of the stiffener are variant.

The final step is to choose a variant parameter, and to adjust it in such a way that the model becomes valid. In case there are multiple variant parameters, one of them is arbitrarily chosen to be adjusted. If the system cannot find a value for that parameter such that the model becomes valid, then another
variant parameter is chosen. This could, for example, be the case if dimension constraints exist that limit the value of the chose parameter. Here, the width of the stiffener cannot be adjusted such that the model becomes valid again, because the width can only be increased, thus resulting in even less space for the handle bar. Therefore the width of the base is adjusted.

In order to find the new value for the width of the base, the width1 distance constraint is temporarily removed, and the width2 distance constraint of the rib temporarily added to the constraint model, in order to be able to update the value of width1 based on the value of width2, i.e. distance2. Then, the algebraic solver is used to update distance2 based on the value of distance3. After that, the geometric constraint solver is used to determine the new value of width1, i.e. the width of the base, based on the new value of distance2. Finally, the value of the parameter of the width1 distance constraint is updated, the width1 constraint is added to the constraint model again, and the width2 distance constraint is removed again.

**Adjustment strategy for interaction constraints**

The adjustment strategy for interaction constraints differs from the adjustment strategy for geometric, algebraic and dimension constraints in that it does not use a dependency graph. Such a graph cannot be built for interaction constraints, because the dependencies between the shapes of the features that are
Automatic model adjustment

created by an interaction constraint on one of the features, are too complex for this. The dependencies between a feature on which an interaction constraint has been specified and other features in the model, depend, among others, on the position and orientation of these features.

In order to convert an invalid model into a valid model, here parameter space sampling is used. The parameter space sampling approach has been inspired by the well-known configuration space approach (Lozano-Pérez 1983) for solving spatial planning problems. A characteristic example of such a problem is to find room for another suitcase in the trunk of a car, which is similar to the problem of finding a place for a feature in a model such that it does not overlap with other features. Instead of a configuration space, the parameter space of the feature model is used to find a solution for the problem.

The parameter space of a feature model is here defined as an \(n\)-dimensional space, with one dimension for each variant feature parameter. Each point in parameter space represents a model that is derived from the original model, i.e. the model specified by the user, by changing one or more of its variant parameters. Some regions of the parameter space represent valid models, other regions represent invalid models. In general, the boundaries between the regions in parameter space that represent valid models and the regions that represent invalid models cannot be analytically determined.

An example of the parameter space for a feature model will be given here. The feature model consists of a rectangular base feature with a trapezoidal rib on both sides, a conical hole with a fixed top radius and major angle through the base and the ribs, and four cylindrical holes. The model contains interaction constraints that specify that the side face of the conical hole should be completely present in the model. It is invalid, because part of the side face of the hole is missing. The model has two dimensions that are variant: the width of the base and the height of the bottom rib (see Figure 7.7(a)). Because the model has two variant feature parameters, its parameter space is two-dimensional (see Figure 7.7(b)). The region of the parameter space that represents models in which all constraints are satisfied, is shaded. It is limited on the left because the base should have a certain minimal width, and at the bottom because the height of the rib should be larger than 0 cm.

![Figure 7.7](image)

**Figure 7.7:** The invalid original model (a), and the parameter space of the model (b).
Given an invalid model and the point in parameter space that represents this model, another point in parameter space is searched that represents a valid model. Preferably, the difference between the invalid and the valid model should be as small as possible. However, because the boundaries of the regions in the parameter space that represent valid models cannot be analytically determined, the valid model that requires the least changes with respect to the invalid model cannot be analytically determined either.

Therefore, in order to find a valid model, samples are created in the parameter space. For each sample, first the variant feature parameters are updated, after that the geometric and algebraic constraints are solved and the new geometry is generated, and finally the other constraints in the model are checked to determine whether the model satisfies all constraints. If so, a valid model has been found, and the model has been adjusted successfully. Otherwise, a new sample is created.

Because this algorithm is rather time consuming, the user is involved if no valid model has been found after a pre-defined number of samples. In such case, the user has to adjust the model to make it valid again.

In order to increase the probability of automatically finding a point in the parameter space that represents a model that satisfies all constraints, a Monte Carlo technique is used to create the samples. Monte Carlo techniques, in general, reduce the number of samples needed to approximate a certain property with a certain accuracy, or increase the accuracy of the approximation with the same number of samples (Fishman 1996). The technique is used here to increase the probability of finding a model that satisfies all constraints, if such a model exists, and thus to reduce the probability that the user has to be involved.

Furthermore, in order to avoid unnecessary large changes to the model, the parameter space is subdivided into subspaces that are subsequently sampled. The subspaces are centred around the point in the parameter space that represents the original (invalid) model. Sampling starts in the subspace that contains that point, and continues in subsequent shells of subspaces around it as shown in Figure 7.8. In each subspace, a specified number of samples are generated.

Figure 7.8: The shells of subspaces in a two-dimensional parameter space that are subsequently sampled.
The probability that a subspace represents at least some models that satisfy all constraints, depends on the relations between the parameters that have been changed from the original model and the unsatisfied constraint, and the extent to which these parameters have been changed. For example, in the model of Figure 7.7(a), the interaction between the conical hole and the base and bottom rib is more likely to be removed by changing the width of the base, than by changing the height of the bottom rib, because the latter does not change the interaction with the base.

Although in this way it is tried to avoid unnecessary large changes to the model, the user may want to fine tune the suggested change, based on the valid model that has been found automatically. This fine tuning can be done with the usual modelling operations.

An example of the use of parameter space sampling to automatically adjust an invalid model, is given here. This example is based on the model of Figure 7.7(a). In order to find a valid alternative model, i.e. a model in which the complete side face of the hole is present, the parameter space of the model is subdivided into subspaces and subsequently sampled.

In the example, the first sample is created in the subspace of the original invalid model; it results in another invalid model. The second sample is created in a subspace on the left of the subspace of the initial model, and results in a model with a smaller width of the base, which is still invalid (see Figure 7.9(a)). The third sample sample is created in a subspace above the subspace of the initial model, and results in a model with a larger height for the bottom rib, which is again invalid (see Figure 7.9(b)). The fourth sample is created in a subspace on the right of the subspace of the initial model, and results in a model with a larger width of the base. This sample represents a valid model (see Figure 7.9(c)); it is accepted and shown to the user.

### 7.4 Automatic model adjustment for feature conversion

In feature conversion (see Section 3.3, Chapter 6 and (de Kraker et al. 1997)), problems occur if a consistent feature model in the other part-oriented view does not exist, because in such a situation feature conversion cannot be successfully completed. The changed feature model then has to be adjusted in order to make a consistent feature model in the other view possible, thus enabling the changed feature model to be converted into a feature model of the other view.

Automatic model adjustment can be used to adjust the feature model of a view that has been changed in order to make a consistent feature model in the other view possible. An example of this is given here, based on the feature model of the part detail design view of a product that is shown in
Figure 7.9: Three samples in the parameter space of Figure 7.7(b) with the associated model geometry; only the last one is valid.

Figure 7.10(a), which is similar to a product from the “NIST Design, Planning, and Assembly Repository” (Regli and Caines 1996). The product does not satisfy all constraints of the part manufacturing planning view, in particular the three protrusions in the front are too close. Fortunately, the length of the two protrusions on the left and right, and the width of the third protrusion in the front are variant.

Figure 7.10: A part detail design view of a product (a) and an incomplete part manufacturing planning view for it (b).
Because the product does not satisfy all constraints of the part manufacturing planning view, a complete, i.e. consistent, feature model for this view cannot be derived (see Figure 7.10(b)). In particular, no slots can be created between the protrusions. Dependency analysis can be used to automatically update the feature model of the part detail design view, in order to satisfy the constraints of the part manufacturing planning view, and to allow a consistent feature model for this view to be derived. Such analysis will first be performed for the distance between the protrusion on the left and the one in the front, and subsequently, if still necessary, for the distance between the protrusion on the right and the one in the front.

During dependency analysis for the distance between the protrusion on the left and the one in the front, the dependency graph of the model is created. The information from this graph, as far as it involves the distance between the two protrusions, is shown in Figure 7.11. Based on this information, the system changes the variant width of the protrusion in the front. This change allows both slots to be created in the part manufacturing planning view, so a second dependency analysis is not necessary.

Figure 7.11: Five feature parameters determine the distance between the protrusions.

The resulting feature models for the part detail design view and the part manufacturing planning view are shown in Figure 7.12.

Figure 7.12: The adjusted model of the part detail design view (a) and the corresponding model of the part manufacturing planning view (b).
7.5 Conclusions

This chapter has presented an approach to automatically adjust non-critical parameters of a form feature model that has become invalid after a modelling operation, in order to make it valid again. In addition, it has been shown that the approach can also be used in consistency maintenance. In that context, it can automatically adjust the feature model of the part-oriented view that has been changed, but for which no consistent feature model in the other part-oriented view exists, in order to make a consistent feature model for that view possible.

The presented approach has been built on top of existing constraint solvers and checkers, which are a geometric and an algebraic constraint solver, and a dimension and an interaction constraint checker, and therefore can support several types of constraints. Although the approach is currently not complete, because it can only automatically adjust dimension parameters of form features, and only supports non-cyclic dependencies between geometric and algebraic constraints, it is able to deal with many situations occurring in practice.

Although automatic model adjustment has been developed to adjust a form feature model in order to make it valid or to make a consistent feature model in another view possible, it can also be applied in other situations, e.g. in the context of families of products.

A family of products is here defined by a prototype feature model and a set of parameters that are related to parameters of features in the model or to parameters of model constraints. Members of a family of products can be created by specifying values for the parameters of the family (Bidarra and Bronsvoort 2000a). Parameters of a member of a family of products may be marked variant if they are regarded non-critical. In addition, parameters of features in the prototype feature model that are not dependent on the parameters of the family may also also be marked variant if they are non-critical for the family of products.

A product is a member of a family of products if it satisfies all constraints specified in the prototype feature model that defines the family. Not every combination of values for the parameters of a family of products results in a member of the family. If the specified values do not result in a member of the family, some of the values of the parameters of the family may be manually adjusted by the user. However, the concept of automatic model adjustment may be applied here as well. The values of the variant parameters of the family, and the variant parameters of the features in the prototype feature model, may be used for this.

Possible future extensions to the automatic model adjustment approach are the following.
The user now still has to specify the value of variant feature parameters. If a method could be devised that is able to automatically generate useful default values for the variant feature parameters, then the user could even be relieved of this.

In addition, the adjustment strategy for interaction constraints could be made more efficient by incorporating more domain knowledge. For example, if information on the interacting shapes would be used in the adjustment strategy, then it could be used to predict the regions in parameter space that represent valid and invalid models, and less samples in parameter space might be needed.

Although automatic model adjustment has only been implemented to support situations that involve part-oriented views, it can also be useful in situations that, in addition, involve the conceptual design view and/or the assembly design view. For example, it could be used to adjust the feature model of a part detail design view, in case a consistent feature model for the conceptual design view is not possible after a change to the feature model of that part detail design view, and to adjust the feature model of the assembly design view, in case a consistent feature model for the manufacturing planning view on a part is not possible after a change to the feature model of the assembly design view.

However, it will probably never be used to adjust the feature model of the conceptual design view in case a consistent feature model for another view is not possible, because the feature model of the conceptual design view normally overrules everything else.
Chapter 8

Example modelling session

This chapter describes a modelling session in which part of the development of a modern version of the historical high-wheel bicycle is performed. It is intended as an example of the use of the enhanced multiple-view feature modelling approach for the development of a non-simple product.

Sections 8.1 to 8.3 give an impression of the way the approach supports the initial design of a product. Section 8.4 shows the refinement of the geometry of a part and the propagation of changes in the feature model of the part detail design view to the feature models of the other views. Section 8.5 gives an impression of adding requirements to the conceptual design view and how the consequences are dealt with. Finally, Section 8.6 illustrates changing the feature model of the assembly design view and propagating these changes to the feature models of the other views.

8.1 Initial conceptual design

In the conceptual design phase, the model of the bicycle is built from conceptual components and interfaces. Conceptual components represent functional elements and consist of a base shape and zero or more shape concepts and references. Interfaces represent the freedom between the components.

The creation of a conceptual component is shown for the front fork of the bicycle. The front fork is used to hold the front wheel. A conceptual component with its shape concepts is specified in three steps.

First, a new component has to be created and its name has to be specified. This results in a component with an empty base shape, and no shape concepts or references.

Second, the geometry of the base shape has to be specified from individual shapes and constraints. For each shape, its nature and type, the values of its dimensions, and the way it is attached have to be specified. Additional constraints may be specified to further specify the relative position of the shapes. Figure 8.1 shows some stages of the creation of the base shape of the front fork component.

Finally, one or more shape concepts may be specified on the base shape of the component. For each shape concept, its class and type of shape (see Figure 8.2(a)), the way it is attached (see Figure 8.2(b)), and values for the parameters of the shape concept (see Figure 8.2(c)) need to be specified.
Figure 8.1: New shapes are attached to the base shape (a) by specifying the nature and the shape type (b) and the attach face (the system initially positions a shape near the centre of the attach face) (c), and are further positioned by manually adding constraints (d,e,f), resulting in the final base shape of the component (g).

Figure 8.2: A shape concept is created on the front fork component (a-c) and shown in a table camera (d) and a geometry camera (e).
The creation of an interface is shown for the interface between the front wheel and the front fork. The interface is of type hinge, i.e. it specifies the position of the wheel with respect to the fork, and allows the wheel to rotate with respect to the fork, but prohibits all other motion between the components. The hinge interface is created in two steps.

First, references have to be created on the components that are to be related by the interface. Because these references are intended for an interface between components, they are named after the interface. The steps to create a hinge reference on the front fork are shown in Figure 8.3.

![Figure 8.3: A reference is created by specifying its class (a) and positioning it (b).](image)

Second, the interface itself is created between the created references on the components. The steps to do this are shown in Figure 8.4. The class of an interface is denoted in the GUI by means of the same icon that is used to represent the interface in a graph camera (see Figure 4.11).

![Figure 8.4: An interface is created by specifying its type (a) and the references (b).](image)

The other interfaces between the components of the bicycle are created in the same way, and are shown in Figure 8.5(a). The complete bicycle configuration is shown in Figure 8.5(b).
Figure 8.5: An overview of all interfaces between the components of the bicycle (a) and the complete bicycle configuration (b).
8.2 Initial part design

In the initial part design phase, the global aspects of the final geometry of the parts that implement the conceptual components, are specified using the part detail design views. The resulting geometry can be used to specify connections in the assembly design view. It can be refined later on.

During the specification of the geometry of a part, the manufacturability of the part is checked by propagating all changes to the manufacturing planning view on the part. If a feature model of the product in that view is not feasible because, for example, the part cannot be manufactured or is very expensive to manufacture, then it might be necessary to split the part into multiple parts that together implement the conceptual component.

An example of a conceptual component that cannot be implemented by a single part, if milling would be the only manufacturing technique available, is the front fork, because it would be too expensive to manufacture. Manufacturing the front fork as a single part from a stock of material by milling would require an enormous amount of material to be milled, in order to fit the wheel in the fork and the fork in the frame, as shown in Figure 8.6(a). However, the front fork can also be composed of three separate parts, each of which can be easily manufactured. Figure 8.6(b) shows the three resulting parts.

![Figure 8.6](a) (b)

Figure 8.6: Material that would need to be removed if the front fork would be made as a single part (a), does not need to be removed if the front fork is composed of multiple parts (b).
8.3 Initial assembly design

In the assembly design phase, a model of the bicycle is built from assembly components, based on the parts that have been specified during the initial part design. Connection features are created between the components, which specify the freedom reduced between the components and the way this reduction is implemented. The assembly that results after creating connection features between components can be encapsulated into a compound component, which can in turn be used as a component in an assembly.

The specification of a connection feature, the encapsulation of the resulting assembly into a compound component, and the use of this compound component in another assembly will be shown here, based on the components of the front fork.

The left leg of the front fork is connected to the crown, using a fix rectangular pen-hole connection feature. The connection feature is created by first creating a rectangular pen form feature on the leg and a rectangular hole form feature in the crown, and, after that, specifying its parameters. Since the geometry of the rectangular pen form feature is already available in the leg component, this part of the geometry of the component is used for the rectangular pen form feature, as shown in Figure 8.7(a). On the other hand, because the geometry of the rectangular hole form feature does not yet exists in the crown component, the geometry of the component is adjusted and the associated part is updated accordingly, as shown in Figure 8.7(b).

![Figure 8.7: A form feature on a component may overlap with the geometry of the part, in which case the geometry of the part need not to be updated (a); otherwise the form feature on the component adjusts the geometry of the components, in which case the geometry of the part has to be updated (b).](image)

Before the finally resulting front fork is in turn connected to other components of the bicycle, it is first encapsulated into a compound component, thus abstracting from the connection features between the sub-components of the front fork. After that, the front fork is connected to the steer component, which had already been connected to the frame. The geometry of this connection and the hierarchical graph of the resulting assembly are shown in
Figure 8.8. Notice that the compound front fork component only contains the
form features of the connections on the compound component, i.e. it contains
no form features for the connections between its subcomponents.

Figure 8.8: The geometry of part of the bicycle assembly (a) with the associated hier-
archical graph (b).

New assembly components and parts have to be associated with the concep-
tual component that they implement, and aspects of their geometry have to be
associated with shape concepts on these components. In case of the front fork,
the three components and their parts are associated with the conceptual com-
ponent front fork, and some form features of the components and the parts
are associated with a shape concept on the conceptual component.

Two examples of the specification of such associations are given here. The
first example shows the creation of a component association between the con-
ceptual component front fork and the assembly design component crown us-
ing the component association menu (see Figure 8.9(a)). The second example
shows the creation of a shape concept - form feature association between the
passage shape concept on the front fork that should position the front wheel
and a hole form feature in the assembly design view that is used to connect
the wheel-axis component to the left leg component (see Figure 8.9(b)).

8.4 Part design: refinement

In the refining part design phase, the geometry of a part can be further refined
using the part detail design view and the part manufacturing planning view.
Each change to the part is checked against the conceptual design view and the
assembly design view, in order to check whether the adjusted geometry still
satisfies the requirements from the conceptual and assembly design view.
As an example of refining part design, the geometry of the crown of the front fork is changed to remove sharp edges and to reduce weight. The resulting part detail design view, assembly design view, and part manufacturing planning view on the corresponding component are shown in Figure 8.10.

Figure 8.9: The creation of a component association (a) and a shape concept-form feature association within it (b).

Figure 8.10: The resulting feature model of the crown in the part detail design view (a), the assembly design view (b), and the part manufacturing planning view (c).
8.5 Conceptual design: adding requirements

After the initial design of the front wheel, the designer notices that it would weight approximately 8 kg, and remembers that a front wheel of a bicycle should be less than 5 kg, to allow a cyclist to manage the bicycle well. Therefore, an additional requirement is added to the conceptual design view that specifies the weight of the front wheel to be less than 5 kg.

The modelling system subsequently detects that the current weight of the front wheel part is too high and pops up the inconsistency browser window to inform the designer on the problem. In order to reduce the weight of the front wheel and thus remove the inconsistency, the designer first creates spokes in the wheel, and after that he reduces the diameter of the centre in the part detail design view (see Figure 8.11).

![Figure 8.11: To reduce the weight of the initial design of the front wheel (a), spokes are created (b) and diameter of the centre is reduced (c).](image)

These changes need to be propagated to the other feature models in order to keep all feature models consistent. The first change only has to be propagated to both the feature model of the manufacturing planning view on the part and the feature model of the assembly component that is associated with the part. The second change, on the other hand, also needs to be propagated to the feature model of the axis component that is connected, because it involves a form feature of a connection feature between the wheel and the axis. In addition, it even needs to be propagated to the left and right leg component, since the form feature on the axis component is shared with the connection features between the axis component and the leg components. Figure 8.12 shows how the change to the axis component is propagated to its associated part, and to the leg components to which the axis is connected, and the parts that are associated with them (only one of the leg components is shown).
8.6 Assembly design: changing the model

After the initial assembly design, the designer realizes that the use of a rectangular pen-hole connection to fit the saddle to the frame causes the assembly of the saddle and the frame to be unnecessarily expensive. The designer realized that it would be much cheaper to use a cylindrical pen-hole connection, because it needs less accurate positioning of the saddle with respect to the frame, and it eases clamping of the saddle. The designer therefore replaces the rectangular pen-hole connection feature with a cylindrical pen-hole connection feature. Figure 8.13 shows the involved components before and after this replacement.
After that, the modelling system checks the consistency of the feature model of the assembly design view and the feature models of the part detail design views on the associated parts. It detects inconsistencies between the new feature models of the components and the feature models of the detail design views on the associated parts. To make the parts and the components consistent again, it replaces the rectangular form features in the feature models of the detail design views on the associated parts with their cylindrical equivalents.

Finally, the modelling system detects an inconsistency between the updated feature models of the part detail design views and the feature models of the manufacturing planning views on the parts. The system therefore also updates the feature models of the manufacturing planning views to make these consistent with the feature models of the part detail design views, and thus with the assembly design view. Figure 8.14 shows the original and updated feature models of the detail design view and manufacturing planning view on the part of the frame that is involved in the connection.

Figure 8.14: A change to the feature model of the part detail design view (a) is propagated to the feature model of the manufacturing planning view (b).

Summarising this chapter, it has been shown how the enhanced multiple-view feature modelling approach can effectively support the development of a non-simple product.
Chapter 9

Conclusions

To conclude this thesis, for the three most important aspects of the enhanced multiple-view feature modelling approach, a summary and conclusions are given, and possible future extensions are described.

9.1 Views

A conceptual design view and an assembly design view have been developed, and integrated with the existing part detail design and part manufacturing planning views.

The conceptual design view supports configuration design of a product. The view adheres to the least commitment principle to allow a designer to specify what is required only. A product is here specified from components with shape concepts and references on them and interfaces between them, and thus allows the designer to specify the way the product is built from functional components and to specify the relations between these components.

The assembly design view supports assembly design of a product. A product is here specified from assembly components with form features on them and connection features between them. An assembly component contains form features for the connection features on it only. Because of this, the assembly design view allows the designer to focus on the connections between the components and the aspects of the geometry of the components that are involved in these connections.

The conceptual design view and the assembly design view have been integrated with the part detail design views and the part manufacturing planning views on the associated parts. An integrated product model has been developed for this. The model consists of a feature model for the conceptual design view, a feature model for the assembly design view on the product, and for each part in the product a feature model for the detail design view and the manufacturing planning view on the part.

The geometry of the assembly design view on a product and the views on each part in the product, is represented by cellular models. A cellular model allows incremental updates and avoids the need to have separate models to represent the geometry of different feature models on the same part or component.
The successful development of the conceptual design view and the assembly design view has shown that the view concept is also feasible in situations with abstract models and/or situations with multiple parts and relations between them. Once again, the advantages of having separate views for separate product development phases emerged. Designers in a particular development phase are no longer bothered by aspects of the model that result from other development phases, but can fully concentrate on the aspects that are important in their phase.

The developed integrated product model can effectively support both the feature models of the newly developed aggregate-oriented views, and the feature models of the existing part-oriented views. It has been shown that it supports techniques that help to guarantee the consistency of the different views well.

Some future extensions to the views are feasible.

First, it would be valuable to extend the conceptual design view with support for more functional information. For example, if the conceptual design view would also support information on the load on a product, then it could be used to check whether the resulting product is strong enough. Currently, the conceptual design view supports only information on the volume, material and weight of a component. The conceptual design view can easily be extended to support more functional information.

Second, it would be valuable to extend the assembly view with support for assembly components at generic level, because this would enable a considerable reduction of the complexity of the model. A generic level can easily be created on top of the current integrated product model, which is at instance level.

9.2 Consistency maintenance

A consistency maintenance approach has been developed for the conceptual design view, the assembly design view, the part detail design views and the part manufacturing planning views.

To support the new types of features in the new views, such as the shape concepts in the conceptual design view and the connection features in the assembly design view, consistency maintenance has been extended. The new types of features needed to be supported by the consistency maintenance approach to be able to make sure that the feature models of all views remain consistent.

Because of the diverse nature of the new views, an accurate consistency definition between the views could not always be automatically created, and therefore the designer is involved in this. By enabling the designer to specify associations between related entities in different views, in addition to the
associations that are automatically generated by the system, an accurate consistency definition can be created. The designer can specify associations between model elements at any time after the elements to be associated have been created.

Automatic consistency checking algorithms have been designed that check whether the feature models of the views remain consistent after a modelling operation. For each association in the consistency definition, constraints are checked by an algebraic constraint solver. If these constraints are satisfied, then the views are said to be consistent. By performing this task after each modelling operation, it is guaranteed that an inconsistency is detected as soon as it occurs.

Again, because of the diverse nature of the new views, the consistency could not always be automatically recovered. In situations where the consistency could not be automatically recovered, the system provides tips to the designer on the aspects of the feature models that have become inconsistent. Based on the tips, the designer can specify operations to make the models consistent again.

The approach provides maximum flexibility to the designer in recovering the consistency; the consistency does not have to be recovered immediately, but the tips remain until the inconsistency has been removed. This allows the designer to complete a complex change to the model of a view that temporarily causes the feature models of the views to be inconsistent, and to deal with the consistency of the views again only after such complex change.

The developed consistency maintenance approach for the conceptual design view, the assembly design view, the part detail design views and the manufacturing planning views has shown that consistency maintenance can profitably be used to integrate aggregate-oriented and part-oriented views of different levels of abstraction. The approach minimises the user interaction that is needed to maintain the consistency of the views, by consulting the user only if the model does not contain enough information to automatically maintain the consistency.

The developed consistency maintenance approach is very flexible with respect to the order in which the development phases have to be performed. The approach supports a designer to start specifying a model of a product in an arbitrary view, and continue with the model for any other view, as preferred. Because of this, it supports top-down as well as bottom-up development of products. In top-down development, a designer probably starts in the conceptual design view, and in bottom-up development in the detail design views on the parts.

A valuable extension would be an approach to automatically associate features in the conceptual design view with features in the assembly design and part detail design views.
9.3 Automatic model adjustment

The automatic model adjustment approach has been developed to automatically adjust an invalid feature model, in order to make it valid again, and to automatically adjust the feature model of a view that has been changed if no consistent feature model in another view is possible, in order to make a consistent feature model in that view possible.

Frequently, only some of the dimensions of an instance of a feature have a value that is critical for the function of a product, whereas others can be changed within a given range without influencing the functionality of the product. To indicate that a dimension of a feature is non-critical, the designer can mark it variant. The automatic model adjustment approach will only adjust dimensions that have been marked variant.

The developed automatic model adjustment approach has proven to be able to automatically make invalid models of the part detail design view and the part manufacturing planning view valid again. This capability is a valuable extension of the semantic feature modelling approach, since it avoids the need for the designer to deal with invalid situations that only need adjustment of a non-critical dimension to make a feature model valid again.

It has also shown to be able to automatically adjust the feature model of a changed view for which no consistent feature model in another view is possible. This is especially useful because it avoids the need to explain problems in the latter view to the designer in the first view, using terms of the first view.

Although automatic model adjustment had been developed to be used in the contexts of design by features and of consistency maintenance, it appeared that it could also be useful in the context of families of objects. In the latter context, it can be used to automatically convert a product that is created from the definition of a family of product, but that is not a member of the family, into a member of that family.

It would be valuable to introduce automatic model adjustment for feature models other than form feature models, such as the feature model of the assembly design view. This could be used if no valid feature model for the detail design view on the part of one of the components of an assembly is possible that is consistent with the feature model of the assembly design view. In such situation, it would be useful to automatically adjust the feature model of the assembly design view to make a consistent feature model in the part detail design view possible.

To allow automatic adjustment of the feature model of the part detail design view if it is not consistent with the feature model of the conceptual design view, a technique would be useful that can automatically generate the shape of the product from a functional description that is specified in the conceptual design view. Deriving the shape of a product from a functional description
of the product is a complex problem that has not been fully elaborated yet. Although a number of function-to-shape approaches has been described in literature, e.g. by Gardan et al. (2001), none of them can automatically generate a shape for a product based on a description of its function without using domain-specific knowledge. Therefore none of the approaches is applicable in a general-purpose multiple-view feature modelling system as described here.

9.4 Concluding statement

To conclude, up to now multiple-view feature modelling approaches were merely multiple-view form feature modelling approaches, and therefore only applicable to support the later phases of the product development process. This thesis has shown that a multiple-view feature modelling approach can also support the earlier phases of the product development process, with views that can deal with preliminary geometry and multiple components.
Bibliography


125


Index

a
aggregate-oriented view, 21
algebraic constraint solver, 91
assembly, 59
assembly component, 56
compound, 56
single, 56
assembly design, 17
assembly graph, 59
assembly model
assembly, 59
component, 56
connection feature, 58
association, 74
automatic model adjustment, 92
strategy, 95, 97, 99

b
base shape, 41

b
base shape, 41

function description, 43
physical attributes, 42
conceptual design, 15, 35
conceptual model
compound, 41
configuration, 44
interface, 44
reference, 44
shape concept, 43
concurrent engineering, 1
configuration, 41
configuration graph, 45
connection feature, 58
consistency maintenance, 2, 22, 69
consistency checking, 33, 85
consistency condition, 32
consistency definition, 73, 81
consistency recovery, 33, 86
constraint checker
dimension, 91
interaction, 91
constraint solver
algebraic, 91
geometric, 91
degrees of freedom, 35, 95
degrees of freedom analysis, 91
DeltaBlue, 92
dependency analysis, 95
dependency graph, 95
design by features, 5
Design For X(DFX), 1
dimension constraint checker, 91
Index

**f**
- feature, 2
- feature class, 19
  - element, 19
  - interface, 19
  - validity condition, 19
- feature conversion, 6
  - multiple-way, 9
  - one-way, 7
- feature dependency graph, 26
- feature library manager, 22
- feature modelling, 1
- feature recognition, 5
  - automatic, 6
  - interactive, 6
- form feature class, 22
- function description, 43

**g**
- GEMCON, 37
- geom, 91, 95
- geometric constraint solver, 91
- geometry camera, 28, 46, 61
- GeoNode, 10, 35
- graph camera, 28, 45

**h**
- hierarchical graph camera, 62

**i**
- inconsistency browser, 78
- interaction constraint checker, 91
- interface, 44
- interface - connection feature association, 76
  - checking, 78

**l**
- link, 32
- link constraint, 83, 85

**m**
- model constraints, 23
- model validation scheme, 89
- Monte Carlo technique, 101
- multiple-view feature modelling, 1
- multiple-way feature conversion, 9

**n**
- non-critical dimension, 93

**o**
- one-way feature conversion, 7
- owner list, 27

**p**
- parameter association, 75
  - checking, 77
- parameter space, 100
  - sampling, 100
- part detail design, 18
- part manufacturing planning, 18
- part-oriented view, 21
- persistent naming scheme, 57
- persistent substitute, 57
- physical attributes, 42
- Pro/Engineer, 24
- PRODES, 37
- propagation of changes, 83

**r**
- reference class, 24, 44
- reference geometry, 57
  - persistent substitute, 57
- relational graph camera, 62
- requirement, 1
  - priority, 70

**s**
- sequential engineering, 1
- shape concept, 43
- shape concept - form feature association, 74
Index

checking, 77
simultaneous engineering, 1
single component, 56
SkyBlue, 91
solving scheme, 90

t
table camera, 47
tip, 78

u
user-defined variable, 23

v
validity checking, 31
validity maintenance, 31
validity recovery, 31
variant feature parameter, 93
view, 1
volumetric cell, 27
Summary

Multiple-view feature modelling
with model adjustment

Alex Noort

Multiple-view feature modelling is a recently introduced approach to product development, which combines concurrent engineering and feature modelling. It supports applications from various phases of product development, by providing an own interpretation of, or view on, a product for each of these applications. Each view has its own feature model of the product. The approach can lead to a higher quality of products in less time, which is one of the most important goals of contemporary product development.

Current approaches to multiple-view feature modelling still have at least three major shortcomings. First, they focus on the later product development phases, in which the geometry of the product has to be fully specified. Second, they deal with single parts only, whereas real product rarely consists of a single part. Third, they discard the possibility that a feature model of a product for some view is invalid or that a consistent feature model in another view cannot be created.

In this thesis, a new approach to multiple-view feature modelling is described that overcomes these shortcomings. It supports high-level product design and design of products with multiple parts, in particular conceptual and assembly design. This includes consistency maintenance for the involved feature models. Further, it supports automatic adjustment of form feature models that are invalid or for which no consistent form feature model in another view can be created.

The conceptual design view allows the designer to determine the configuration of a product by specifying components, which are to be implemented by one or more parts, and interfaces between them, which are to be implemented by a connection. Components are specified by means of a base shape with shape concepts and references on it. The base shape is used to attach and position the shape concepts and references, and gives an impression of the shape
of the component. Shape concepts are used to specify functional requirements on the geometry of a component, e.g. that there should be a passage, possibly with block shape, somewhere on the component. References are used to position interfaces between components, shape concepts and other references. Interfaces between components are specified by means of Degrees Of Freedom (DOFs) between references on components.

The assembly design view focuses on connection design, and allows the designer to specify the type of connection between components and the geometry for the connection on the components. A component can be a single component, which represents a single part, or a compound component, which represents multiple connected sub-components. The geometry of components only needs to be specified as far as it is involved in connections. Connections are specified on regions of the geometry of the connected components, such as a rib and a slot, and represent a reduction of the DOFs between the components.

Consistency maintenance integrates all views with each other, by ensuring that their feature models represent the same product or part of it, i.e. that their feature models are consistent. It checks the consistency of pairs of feature models, based on the consistency definitions for those pairs, and if an inconsistency is found, it recovers the consistency of the feature models. To optimise the performance and to reduce the number of consistency maintenance techniques, consistency maintenance is only performed for a minimal set of pairs that together allow all views to be kept consistent.

Automatic model adjustment is able to automatically adjust the model of a product in case the feature model of one of its views has become invalid, or a consistent feature model in another view cannot be created. In current feature modelling systems, the model of the product has to be manually adjusted in order to make an invalid feature model valid again, or to make a consistent feature model in another view possible. The main disadvantage of manually adjusting the model of a product is that it can be a lot of tedious work for a designer to get a model that satisfies all requirements on the product at the same time. Automatic model adjustment only adjusts dimensions in the feature model that are not critical for the function of the product. It has been built on top of constraint solvers for geometric and algebraic constraints and constraint checkers for dimension and interaction constraints.

This thesis shows that a multiple-view feature modelling approach can also support the earlier phases of the product development process, by describing views that support conceptual and assembly design, and their integration with views that support part detail design and manufacturing planning. In addition, it shows that automatic model adjustment is a feasible and useful technique.
Samenvatting

Feature modelleren met meerdere interpretaties en modelaanpassing

Alex Noort

Feature modelleren met meerdere interpretaties is een methode voor de ontwikkeling van produkten die recentelijk geïntroduceerd is en concurrent engineering en feature modelleren combineert. De methode ondersteunt toepassingen uit verschillende produktontwikkelingsfasen met behulp van een gespecialiseerde interpretatie voor elk van die fasen. Elke interpretatie heeft zijn eigen feature model van het produkt. De methode maakt het mogelijk om in kortere tijd betere produkten te ontwikkelen, hetgeen tegenwoordig een van de belangrijkste doelstellingen is bij de ontwikkeling van produkten.

De huidige feature modellerenmethoden die meerdere interpretaties ondersteunen hebben nog minstens drie belangrijke tekortkomingen. Ten eerste beperken ze zich tot de latere produktontwikkelingsfasen waarin de vorm van het produkt al volledig vastligt. Ten tweede ondersteunen ze alleen de ontwikkeling van produkten die uit één enkel onderdeel bestaan, terwijl deze in de praktijk zelden voorkomen. Ten derde bieden ze geen ondersteuning in situaties waarin het feature model van een produkt niet geldig is of een consistent feature model voor een andere interpretatie niet bestaat.

In dit proefschrift wordt een nieuwe methode voor feature modelleren met meerdere interpretaties beschreven die deze tekortkomingen niet heeft. De methode ondersteunt het ontwerpen van produkten op een hoog abstractieniveau en het ontwerpen van produkten die uit meerdere onderdelen bestaan, in het bijzonder conceptueel ontwerpen en het ontwerpen van assemblages. Daarbij wordt de consistentie van de betrokken feature modellen gehandhaafd. Verder ondersteunt de methode het automatisch aanpassen van vormfeature modellen die ongeldig zijn of waarvoor een consistent vormfeature model in een andere interpretatie niet bestaat.

De interpretatie voor conceptueel ontwerpen ondersteunt de ontwerper bij het ontwerpen van de configuratie van een produkt met behulp van componenten, die door een of meerdere onderdelen worden geïmplementeerd, en interfaces tussen de componenten, die door verbindingen worden
geïmplementeerd. Componenten worden gespecificeerd door middel van een basisvorm met vormconcepten en referenties daarop. De basisvorm wordt gebruikt voor het positioneren van de vormconcepten en referenties, en geeft een indruk van de vorm van de component. Vormconcepten worden gebruikt om functionele eisen die gesteld zijn aan de vorm van een component te specificeren, bijvoorbeeld dat er een doorgaand gat in de component moet zitten. Referenties worden gebruikt voor het positioneren van interfaces tussen componenten, vormconcepten en andere referenties. Interfaces tussen componenten worden gespecificeerd door middel van de gewenste bewegingsvrijheid tussen de componenten.

De interpretatie voor assemblage ontwerpen is gericht op het ontwerpen van verbindingen, en ondersteunt de ontwerper bij het specificeren van het type verbinding en de vorm van de componenten ten behoeve van de verbinding. Een component kan een enkelvoudige component zijn, die één onderdeel representeert, of een samengestelde component, die meerdere verbonden onderdelen representeert. De vorm van de onderdelen hoeft slechts te worden gespecificeerd voor zover deze te maken heeft met de verbinding. Verbindingen worden gespecificeerd tussen delen van de vorm van de verbonden componenten, zoals een ribbe en een slot, en representeren een beperking van de bewegingsvrijheid tussen de componenten.

Het proces dat de consistentie van de feature modellen handhaaft integreert de interpretaties met elkaar, door te garanderen dat de feature modellen van al de interpretaties hetzelfde produkt of een deel ervan representeren. Het test de consistentie van paren van interpretaties, gebaseerd op de consistentiedefinities voor de paren, en als er een inconsistentie wordt gevonden zorgt het ervoor dat de feature modellen weer consistent worden. Om de verwerkingssnelheid te optimaliseren en het benodigde aantal technieken te verminderen, wordt slechts de consistentie van een minimaal aantal paren van interpretaties gehandhaafd, zodanig dat alle interpretaties consistent gehouden kunnen worden.

Automatische modelaanpassing maakt het mogelijk om het model van een produkt automatisch aan te passen als het feature model van een van de interpretaties ongeldig is geworden, of een consistent feature model in een andere interpretatie niet bestaat. In de huidige feature modelleersystemen moet het model van het produkt handmatig worden aangepast om een ongeldig feature model weer geldig te maken of een consistent feature model in een andere interpretatie mogelijk te maken. Het grootste nadeel van het handmatig aanpassen van het model van een produkt is dat het voor een ontwerper een tijdrovend werk kan zijn om een model te krijgen dat aan alle eisen voldoet. Automatische modelaanpassing past alleen dimensies van het feature model aan die niet kritisch voor de functie van het produkt zijn. Het maakt gebruik van constraint solvers voor geometrische en algebraïsche constraints en constraint checkers voor dimensie en interactie constraints.
Dit proefschrift heeft laten zien dat feature modelleren met meerdere interpretaties ook de eerdere fasen van het produktontwikkelingsproces kan ondersteunen, door het beschrijven van interpretaties die conceptueel en assembleren ondersteunen, en het beschrijven van de integratie van deze interpretaties met de interpretaties die detailontwerp en fabricagevoorbereiding ondersteunen. Daarnaast heeft het ook laten zien dat autonome modelaanpassing een haalbare en nuttige techniek is.
Curriculum Vitæ

Alex Noort was born on June 13th, 1974, in Rijnsburg, The Netherlands.

In 1992, he received his VWO certificate at the Pieter Groen College in Katwijk. In 1997, he received his master’s degree in Computer Science at Delft University of Technology. His master’s thesis describes a method to solve over-constrained geometric models, and resulted in his first scientific publication. The research was carried out at the Computer Graphics and CAD/CAM group of Delft University of Technology.

After receiving his master’s degree, he started his PhD project in the same group. The project involved multiple-view feature modelling and automatic adjustment of feature models, and resulted in this thesis.

In 2002, he joined the CPB Netherlands Bureau for Economic Policy Analysis in Den Haag, where he can be reached at A.Noort@cpb.nl.